MS2665C/67C/68C
Spectrum Analyzer
Operation Manual
Programming
(PTA Control)

MS2665C/67C/68C
Spectrum Analyzer
Operation Manual

Programming

(PTA Control)

TABLE OF CONTENTS

SECTION 1 GENERAL 1-1

PTA Specifications 1-4

PTL Command of PTA 1-6

External Interfaces of MS2665C/67C/68C 1-11

RS-232C interface 1-11

GPIB interface 1-11

Parallel (centronics) interface 1-12

Screen Configuration of PTA 1-13

Physical screen configuration 1-13

Display form 1-14

SECTION 2 PTA OPERATION 2-1

Outlining the Operation 2-3

Operations Related to PTA Program 2-6

Startup of PTA 2-6

Loading the PTA program from memory card 2-6

Execution, stop of the PTA program 2-7

PTA termination 2-8

Format of PTA program file 2-8

Operations Related to PTA Library 2-9

Loading the PTA library from memory card 2-9

Registering the PTA library to user key 2-11

Execution, stop of the PTA library 2-15

Format of PTA library file 2-15

Operations related to PTA library 2-16

Panel Key Operations during PTA Program/Library Execution 2-17

Data input keys 2-17

Operation of other panel keys 2-18

Menu Construction of the PTA Key 2-19

SECTION 3 PTL COMMANDS 3-1
Program Input Command 3-4
PCOPY Command 3-5
DELETE Command 3-6
RENUM Command 3-7
LIST Command 3-8
LISTG Command 3-9
PMEMO Command 3-10
Immediate Execution Command 3-11
RUN Command 3-12
STOP Command 3-13
CONT Command 3-14
RESET Command 3-15
SAVE Command 3-16
LOAD Command 3-17
OVERLAY Command 3-18
PDEL Command 3-19
PLIST Command 3-20
STARTP Command 3-21
CANCEL Command 3-22
EDITLIB Command 3-23
EDITPTA Command 3-24
RENAME Command 3-25
LIBMEM Command 3-26
SAVELIB Command 3-27

SECTION 4 PTL 4-1

Elements of Statement Configuration 4-3
Line number 4-3
Constants 4-4
Variables 4-6
Multi statement 4-8
Functions 4-9
Arithmetic operators 4-14
Relational operators 4-15
String concatenation (the "+" operator) 4-16
Formats 4-17
Label 4-18

Basic Statements 4-19
Comment (REM statement) 4-19
Array declaration (DIM statement) 4-20
Initialization (CLEAR statement) 4-22
Substitution (LET statement) 4-23
Branch (GOTO statement) 4-24
Termination of execution (STOP statement) 4-24
Branch to subroutines (GOSUB statement) 4-24
Return from subroutines to main routine (RETMAIN statement) 4-25
Return from subroutines (RETURN statement) 4-25
Decision (IF statement) 4-26
Repetitions start (FOR statement) 4-27
Repetition termination (NEXT statement) 4-28
Key-input (INPUT statement) 4-29
Display (PRINT statement) 4-30
Reverse display (PRINTR statement) 4-34
Positioning the cursor (LOCATE statement) 4-35
Data statement (DATA statement) 4-35
Reading data (RDATA statement) 4-36
Read specification of data statement (RESTORE statement) 4-36
Setting measurement parameters (PUT and WRITE 1000 statements) 4-37
Measurement parameter/data read (GET, COM and READ 1000 statements) 4-38
Program loading and execution (CHAIN statement) 4-40
ENABLE EVENT statement 4-40
DISABLE EVENT statement 4-44
ON EVENT statement 4-44

RETINT statement 4-45
Character size specification (DCHSIZE statement) 4-46
Home position (HOME statement) 4-47
Delete (ERASE statement) 4-47
Time wait (WAIT statement) 4-47
System subroutine execution (CALL statement) 4-48
ON ERROR statement 4-48
OFF ERROR statement 4-49
RETERR statement 4-49
RETRY statement 4-50
RESUME statement 4-50
GIVEUP statement 4-51
Error branch (ERROR statement) 4-51
Error main (ERRMAIN statement) 4-52
Data input 1 (READ statement) 4-52
Data input 2 (BREAD statement) 4-53
Data input 3 (WREAD statement) 4-53
Data output 1 (WRITE statement) 4-54
Data output 2 (BWRITE statement) 4-54
Data output 3 (WWRITE statement) 4-55
Data writing to the dual port memory (WDPM statement) 4-57
Data reading from the dual port memory (RDPM statement) 4-57
S.O.S (SOS) 4-58
Setting the pseudorandom number sequence (RNDMIZE statement) 4-59
Calling the PTA library (CALLIB statement) 4-59
Removing the PTA library from program memory (REMOVE statement) 4-60
Clearing common variables (COMCLEAR statement) 4-61
Setting CALLIB parameter values (PARASET statement) 4-61
Loading the PTA library file LOADLIB statement) 4-62

SECTION 5 EXTENDED PTL 5-1

System Variables 5-3

System Subroutines 5-5
CER and CRN subroutines 5-7
CFL subroutine 5-8
DCH subroutine 5-9
DLN subroutine 5-11
DRC subroutine 5-13
DCR subroutine 5-15
DAR subroutine 5-17
DEF subroutine 5-19
OPNI, OPNO and FDEL subroutines 5-20
DALD and DASV subroutines 5-21
CLS subroutine 5-22
IFC subroutine 5-22
OPNITF, OPNOTF, FDEFTF subroutines 5-23
DALDTF, DASATF subroutines 5-24
CLSTF subroutine 5-25
RSV subroutine 5-26
TCT subroutine 5-27
DEV subroutine 5-27
GST subroutine (GST) 5-28
Interface control subroutine (GPIB and RS-232C) 5-29
PNLU and PNLL subroutine 5-31
COPY subroutine 5-32
CONV subroutine 5-33
SWLG subroutine 5-34

System Functions 5-35
MAX function 5-38
MIN function 5-39
BNDL, BNDH, MESL, and MESH functions 5-40
RPL1 and RPL2 functions 5-42
RPL3 function 5-43
PEKL and PEKH functions 5-44
POLL and POLH functions 5-46
PLRH, PLLH, PLRL and PLLL functions 5-48
PFRQ function 5-50
SUM function 5-51
PSML and PSMH functions 5-52
DPOS and DNEG functions 5-54

SECTION 6 REMOTE CONTROL COMMANDS USED WITH PTA PROGRAM/LIBRARY 6-1

Outline 6-3

PTA Dedicated Remote Control Commands 6-4

SECTION 7 EXTERNAL INTERFACE IN PTA 7-1

Outline 7-3

Selection of Controlled Interface Port from PTA 7-4

RS-232C Functions in PTA 7-5

GPIB Functions in PTA 7-7

Function as controller 7-7

Function as device 7-11

Functions of Parallel (centronics) in PTA 7-12

Dual Port Memory 7-13

SECTION 8 PTA ERROR MESSAGES 8-1

Error Message Format 8-4

ERROR Statement 8-5

ERRMAIN Statement 8-6

Error Processing Subroutines 8-7

ON ERROR statement 8-7

OFF ERROR statement 8-7

Returning from error processing subroutines
(RETERR, RETRY, RESUME and GIVE UP statements) 8-7

ERRREAD (m) function 8-8

Error List 8-9

SECTION 1
GENERAL

TABLE OF CONTENTS

PTA Specifications 1-4
- PTL Command of PTA 1-6
- External Interfaces of MS2665C/67C/68C 1-11
- RS-232C interface 1-11
- GPIB interface 1-11
- Parallel (centronics) interface 1-12

Screen Configuration of PTA 1-13
- Physical screen configuration 1-13
- Display form 1-14

SECTION 1
GENERAL

PTA (Personal Test Automation) is the MS2665C/67C/68C spectrum analyzer equipped with a programming language interpreter function to enable programming controls and calculations directly connected with the measurement system with a high-speed language of PTL (Personal Test Language).

In addition to the basic commands similar to BASIC, PTL provides GPIB control commands, file operation commands, screen control commands and function control commands for controlling most functions of the MS2665C/67C/68C.

Programs that can be executed by PTA are two types, including the "PTA program" to be executed by specifying RUN" from the PTA menu, and the "PTA library" to be executed by registering it to the User key menu. Both the PTA program and PTA library are prepared using the universal edit program on an external personal computer, and registered to MS2665C/67C/68C via RS-232C or GPIB. It is also possible to save the edited PTA program/PTA library to a memory card, as a text file, and input it to the memory card interface of MS2665C/67C/68C. Since inputted programs can be stored in the built-in nonvolatile program memory, no efforts otherwise required for reloading after each power-off are necessary.

As for the external interface to PTA, there are GPIB, RS-232C parallel (Centro) and PTA parallel I/O. RS-232C and GIPB connect an external computer to realize PTA to communicate with the computer through a communication memory (dual port memory). PTA is also capable of controlling an automatic inspection system for electronic components or a trimming machine by connecting with such equipment using PTA parallel I/O.

PTA Specifications

The PTA specifications are listed below:

■Display

• Number of displayed characters :	54 characters/line×20 lines (43 characters/line for menu display)
• Displayable characters :	Alphabetic upper-and lower-case characters, numerals, special symbols, and cursors
• Character font :	6×12 dots (small type)
• Graphics :	Straight line, square, circle and arc
• Screen :	320×240 dots×16 screens

■Input and execution control

• Input :	Front panel, and external computer (by RS-232C, GPIB)
• Execution control :	Front panel, and external computer (by RS-232C, GPIB)

■Memory

• Program memory :	196 kbytes
• Memory card :	256 kbytes, 512 kbytes, 1 Mbyte, 2 Mbytes

■Language Version PTL - V1.6

• Commands :	Edit commands Program execution commands File commands
• Statements :	Basic statements GPIB statements Event statements Dual port statements
• Subroutines :	Display subroutines Filing subroutines GPIB subroutines Interface subroutines Panel subroutines Waveform memory subroutines

- Functions : Arithmetic functions
  Boolean functions
  Statistical functions
  Character string functions
  System functions

- Variables : Up to a maximum of 256 user-defined variables
  (numeric variable, character string variable)
  System variables 22 types

- Data types : Real number:
  Significant digits =15 digits
  Exponential = 10^{308} to 10^{-307}
  Integer –32768 to 32767
  Character: 256 characters max.
  Bit: 8 bits max.

■Interfaces

- RS-232C
- GPIB
- Parallel (centronics)
  (option 10)

PTL Command of PTA

Table 1-1 shows the PTL (Personal Test Language) commands provided with the PTA :

Table 1-1 PTL Command of PTA

Item	Format
Edit Commands	
Program input	Line number statement
Copy	PCOPY new start-line number, [increment], copy-source start-line number, copy-source stop-line number
Delete	DELETE [start-line number][,[stop line number]] or [line number] [RETURN]
Renumber	RENUM [new start-line number[,increment[, old start-line number [old stop-line number]]]]
List output (CRT)	LIST [start-line number] [,[stop-line number]
List output (printer)	LISTG address [,start-line number] [,[stop-line number]]
Program size	PMEMO
Execute commands	
Program execution start	[RUN] menu key or RUN [start-line number] [, suspension-line number]
Suspension of program execution	[STOP] menu key
Continuation of suspended program execution	[CONT] menu key, CONT [suspension-line number]
Discontinuation of program execution	[RESET] menu key
Direct execution	Statement [RETURN]
File commands	
Save file	SAVE program name [, start-line number [, stop-line number]]
Load file	LOAD program name
Overlay	OVERLAY
File list display	[PLIST] menu key
Delete file	PDEL Program name
Start-up registration	STARTP program name or STARTP @
Start-up cancel	CANCEL or CANCEL @

Table 1-1 PTL Command of PTA (Continued)

Item	Format
Statements	
Comments	REM ["comment"] or 'comment
Array declaration	DIM array variable
Assignment	[LET] variable = expression (functions,variables or constants)
Jump	GOTO line number or GOTO *label
Jump to subroutines	GOSUB line number or GOSUB *label
Return from subroutines	RETURN
Decision	IF condition statement
Loop beginning	FOR numeric variable = initial value TO ending value STEP step value
Loop end	NEXT numeric variable
Key input	INPUT ["display character string",] variable [, variable...]
Display	PRINT variable [: format][, variable [: format]...][;]
Reverse display	PRINTR variable [: format][, variable [: format]...][;]
GPIB input	READ address, input variable [, variable...]
GPIB input (1 byte)	BREAD address, input variable [, variable...]
GPIB input (2 bytes)	WREAD address, input variable [, variable...]
GPIB output	WRITE address, variable [: format][;]
GPIB output (1 byte)	BWRITE address, variable [: format]
GPIB output (2 bytes)	WWRITE address, variable [: format]
Set measurement parameter	PUT string variable (or string)
Read measurement parameter (1)	GET string variable (or string), input variable
Read measurement parameter (2)	COM character string variable (or character constant)> input variable
Wait	WAIT time (unit is second, minimum 0.01 s.)
Subroutine call	CALL subroutine name
Cursor location (home position)	HOME
Cursor location	LOCATE (X, Y)
Erase screen	ERASE
Program end	STOP
Display error	Line NO␣SOS␣ "Grammer error expression"
Jump on Error	ERROR (error number, line number or * label)
Error main	ERRMAIN
Return to main routine	RETMAIN

Table 1-1 PTL Command of PTA (Continued)

Item	Format
Statement (cont'd)	
Initialization of variable	CLEAR
Data statement	DATA constant [, constant..., constant]
Specification of input data statement	RESRORE [line number or * label]
Data-statement input	RDATA variable [, variable...]
Program reading/execution	CHAIN "file name"
Register an error interrupt routine	ON ERROR line number or * label
Unregister an error interrupt routine	OFF ERROR
Return from an error interrupt routine	RETERR RETRY RESUME line number or * label GIVEUP
Register an event interrupt routine	ON EVENT I/O number, line number or * label
Enable an event interruption	ENABLE EVENT I/O number, event 3, event 2, event 1, event 0
Disable an event interruption	DISABLE EVENT I/O number [, event 3, event 2, event 1, event 0
Return from an event	RETINT
Character size specification	DCHSIZE character size number
Pseudorandom number string setting	RNDMIZE
Dual-port-memory statement	
Write data	WDPM memory No., variable [: format]
Read data	RDPM memory No., input variable

Table 1-1 PTL Command of PTA (Continued)

Item	Format
Screen subroutines (cont'd)	
Screen subroutines	
Displayed-item erasure	CALL CER(M)
Displayed-item restoration	CALL CRN(M)
Screen erasure	CALL CFL(M)
Character string display	CALL DCH(X,Y,text,M[,N])
Straight-line display	CALL DLN(XØ,YØ,X1,Y1,M[,N])
Square display	CALL DRC(XØ,YØ,X1,Y1,M[,N])
Circle display	CALL DCR(X,Y,R,M[,N])
Arc display	CALL DAR(XØ,YØ,RØ,W1,W2,M[,N])
Soft key label registration	CALL DEF(M,text)
Filing subroutines	
Open a file (read)	CALL OPNI Character string variable (or character constant)
Open a file (write)	CALL OPNO Character string variable (or character constant)
Delete a file	CALL FDEL Character string variable (or character constant)
Load data	CALL DALD variable
Save data	CALL DASV variable
Close a file	CALL CLS
Panel subroutines	
Lock front-panel key operation	CALL PNLL(Ø)
Unlock front-panel key operation	CALL PNLU(Ø)
Waveform memory subroutine	
Copy memory	CALL COPY(MØ,M1)
Data conversion	CALL CONV(K,MØ,M1,PØ,P1[,D])
Frequency axis logarithm conversion	CALL SWLG(K,MØ,M1)
GPIB subroutine	
Interface clear (switching to system controller port)	CALL IFC
Service request	CALL RSV(M)
Take controller	CALL TCT(M)
Switching to device port	CALL DEV
Interface subroutine	
Status byte read	CALL GST(port No., address, input variable)
Interface control	CALL GPIB(port No., control item No.)
Function	
Arithmetic functions	SIN, COS, TAN, ASN, ACS, ATN, LN, LOGEXP, SQR, ABS, SGN, INT, ROUND, DIV, FIX
Boolean functions	NOT,AND,OR,EOR
Character string functions	CHR, VAL, HVAL, BVAL, ASC, CHR$, CVI, CVD, MKI$, MKD$, STR$, HEX$, OCT$, BIN$, INSTR, LEFT$, MID$, RIGHT$, STRING$, LEN, SLEN, SGET$

Table 1-1 PTL Command of PTA (Continued)

Item	Format
Function (cont'd)	
Statistical functions	max, min, sum, mean, var, sta
Dedicated functions	ERRREAD, STATUS, DTREAD$, RND
System variable	EX0, EX1, EX2, EX3, EX4, EX5, EX6, DTØ, DT1, DT2, DT3, DT4, XMA, XMB, XMG, XMT, XMT, SMA, SMB, SMT, IMA, IMB, RMA, RMB, IOA, IOB, IOC, IOD, EIO
System function	
Maximum value	MAX (M, PØ, P1, N)
Minimum value	MIN (M, PØ, P1, N)
Frequency measurement 1	BNDL (M, PØ, L, N)
Frequency measurement 2	BNDH (M, PØ, L, N)
Frequency measurement 3	MESL (M, PØ, L, N)
Frequency measurement 4	MESH (M, PØ, L, N)
Ripple 1	RPL1 (PØ, P1, N [,R])
Ripple 2	RPL2 (PØ, P1, N [,R])
Ripple 3	RPL3 (PØ, P1, N [,R])
Peak 1	PEKL (M, PØ, L, N [,R])
Peak 2	PEKH (M, PØ, L, N [,R])
Poll 1	POLL (M, PØ, L, N [,R])
Poll 2	POLH (M, PØ, L, N [,R])
Maximum 1	PLRH (M, PØ, N [,R])
Maximum 2	PLLH (M, PØ, N [,R])
Minimum 1	PLRL (M, PØ, N [,R])
Minimum 2	PLLL (M, PØ, N [,R])
Index point frequency	PFRQ (PØ)
Sum	SUM (PØ, P1, N)
Adding search 1	PSML (M, PØ, L, N)
Adding search 2	PSMH (M, PØ, L, N)
Judgment 1	DPOS (M, PØ, P1, N1, N2)
Judgment 2	DNEG (M, PØ, P1, N1, N2)

External Interfaces of MS2665C/67C/68C

MS2665C/67C/68C provides an RS-232C interface and GPIB interface as standard. In addition, a parallel (centronics) interface (option 10) is optionally available. The usage of these interfaces differs by the setting of the connection port.

RS-232C interface

- When the RS-232C interface is selected as the connection port for the external controller (Connect to Controller):

  Connect the device that controls the spectrum analyzer, for example, a host computer. Execution of the PTA program/PTA library is indicated and the PTA program can be interfaced via the dual port memory. Also, the PTA program/PTA library is registered.

- When the RS-232C interface is selected as the connection port to the printer/plotter (Connect to Printer/Plotter):
  By specifying COPY from the PTA program/library, the printer copies the screen.

- When the RS-232C interface is selected as the connection port to the a peripheral device (Connect to Peripheral):

  Serial data transfer is available between the PTA program/library and the external device.

GPIB interface

- When the GPIB interface is selected as the connection port for the external controller (Connect to Controller): In this case, the GP-IB interface enters the devicxe port state. Connect the device that controls the spectrum analyzer, for example, a host computer. Execution of the PTA program/PTA library is indicated and the PTA program can be interfaced via the dual port memory. Also, the PTA program/PTA library is registered.

- When the GPIB interface is selected as the connection port to the printer/plotter (Connect to Printer/Plotter): By specifying COPY from the PTA program/library, the printer copies the screen.

- When the GPIB interface is selected as the connection port to the a peripheral device (Connect to Peripheral): In this case, the GPIB interface works as a system controller port. It is possible to control external devices from the PTA program/library.

Parallel (centronics) interface

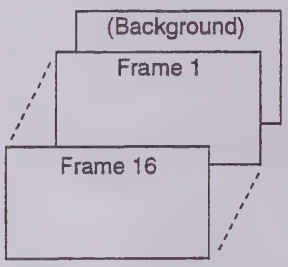
- When the parallel (centronics) interface is selected as the connection port to the printer/plotter (Connect to Printer/Plotter):

  By specifying COPY from a PTA program/library, the printer copies the screen.

Screen Configuration of PTA

This section describes the screen specifications of PTA mounted in the MS2665C/67C/68C.

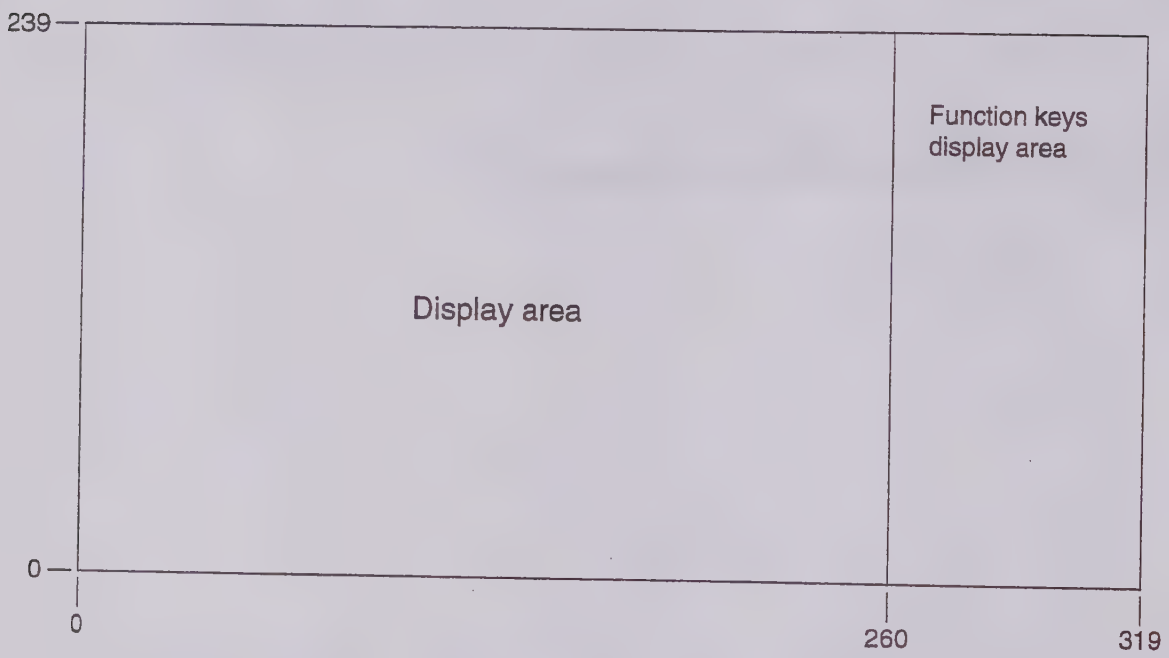
Physical screen configuration

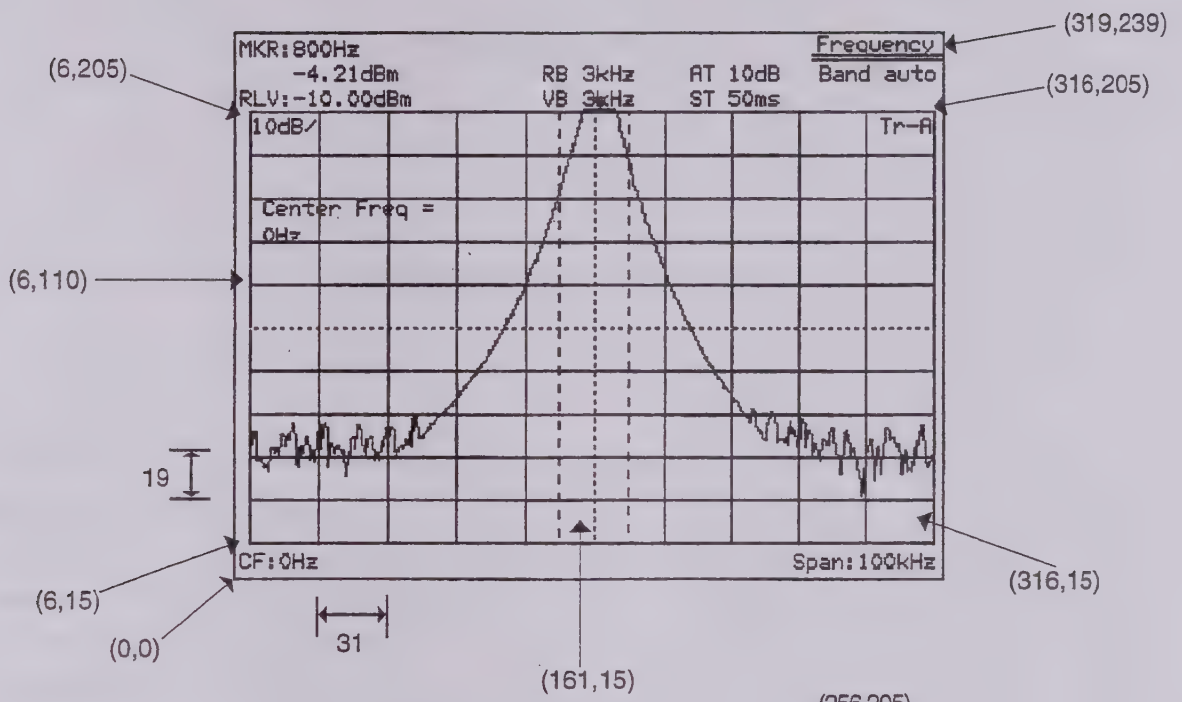


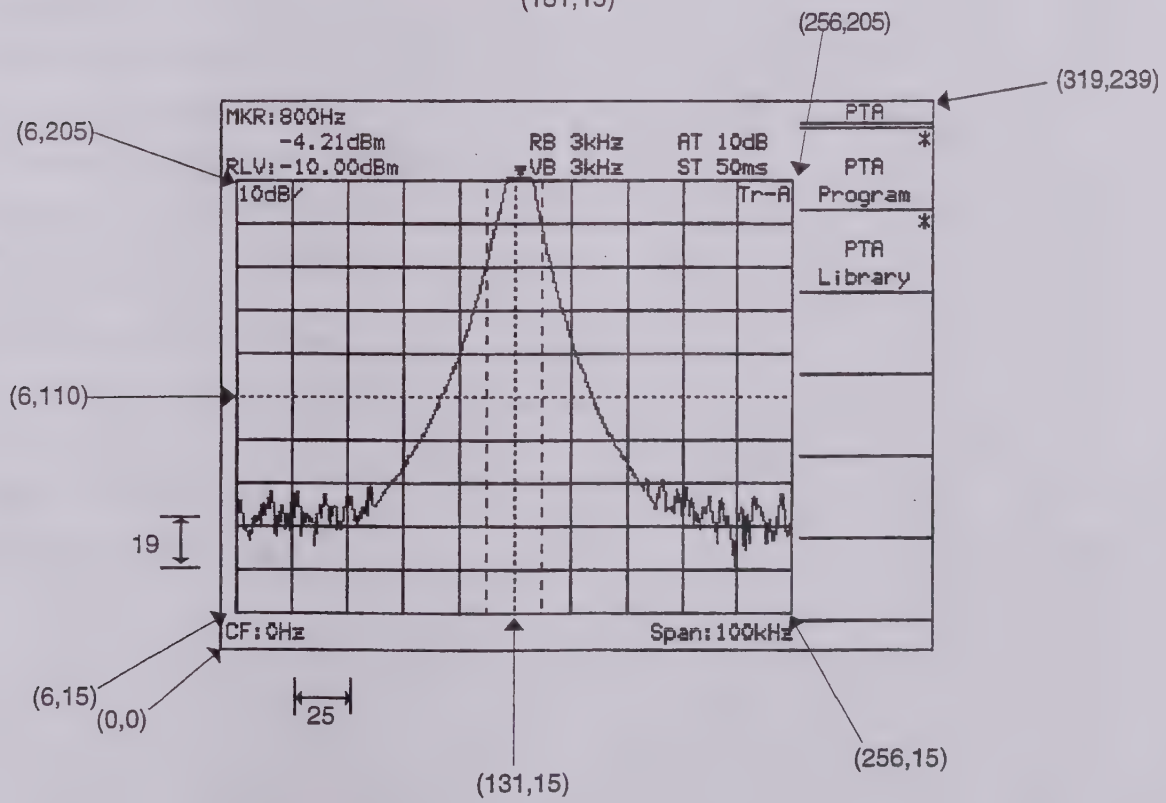
Frame 1 : Waveform display background
2 : Scale lines
3 : Waveform 2
4 : Waveform 1
5 : Parameters (title, reference level, RBW, VBW, center frequency, span, etc.)
6 : Display lines, reference markers
7 : Triggers, indicators
8 : Marker zones
9 : Template/mask standard lines
10 : Multi-marker Nos.
11 : (Not used)
12 : Markers, marker values
13 : PTA screen
14 : Menu background
15 : Menu characters
16 : Setup and parameter characters, error messages

Note: This frame number is controlled inside the spectrum analyzer. The number is different from the number used by screen subroutines such as CALL CFL.

Display form




MKR:800Hz
-4.21dBm
RLV:-10.00dBm
RB 3kHz
VB 3kHz
AT 10dB
ST 50ms
Frequency
Band auto
10dB/
Tr-A
Center Freq =
0Hz
CF:0Hz
Span:100kHz
(6,205)
(319,239)
(316,205)
(6,110)
19
(6,15)
(0,0)
31
(161,15)
(316,15)


MKR:800Hz
-4.21dBm
RLV:-10.00dBm
RB 3kHz
VB 3kHz
AT 10dB
ST 50ms
PTA
PTA
Program
PTA
Library
10dB/
Tr-A
CF:0Hz
Span:100kHz
(256,205)
(319,239)
(6,205)
(6,110)
19
(6,15)
(0,0)
25
(131,15)
(256,15)

SECTION 2

PTA OPERATION

TABLE OF CONTENTS

Outlining the Operation 2-3

Operations Related to PTA Program 2-6

- Startup of PTA 2-6
- Loading the PTA program from memory card 2-6
- Execution, stop of the PTA program 2-7
- PTA termination 2-8
- Format of PTA program file 2-8

Operations Related to PTA Library 2-9

- Loading the PTA library from memory card 2-9
- Registering the PTA library to user key 2-11
- Execution, stop of the PTA library 2-15
- Format of PTA library file 2-15
- Operations related to PTA library 2-16

Panel Key Operations during PTA Program/Library Execution 2-17

- Data input keys 2-17
- Operation of other panel keys 2-18

Menu Construction of the PTA Key 2-19

SECTION 2
PTA OPERATION

Outlining the Operation

PTA of MS2665C/67C/68C is capable of executing/operating two types of automation programs, the "PTA program" and "PTA library".

PTA program :

One program can be loaded and executed on the execution memory (RAM) of MS2665C/67C/68C.

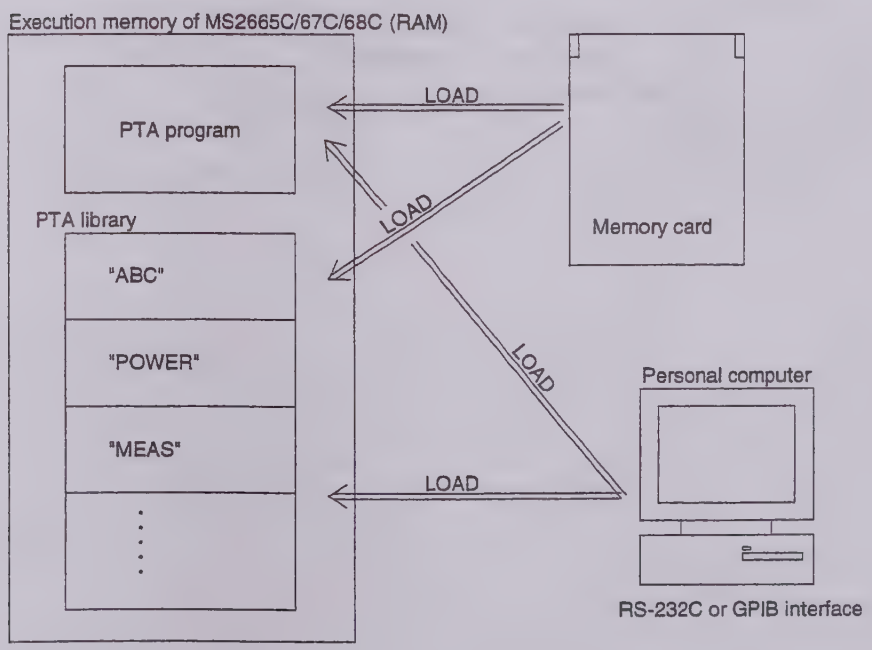
A PTA program is loaded and executed on menus following [SHIFT] + [PTA]→[PTA Program : F1].

This function is the same as the PTA functions and PTA program execution provided in the existing measuring instruments of our make (for example, MS2601B, MS2602A, MS8604A, etc.).

PTA library :

Multiple programs can be loaded and executed on the execution memory (RAM) of MS2665C/67C/68C.

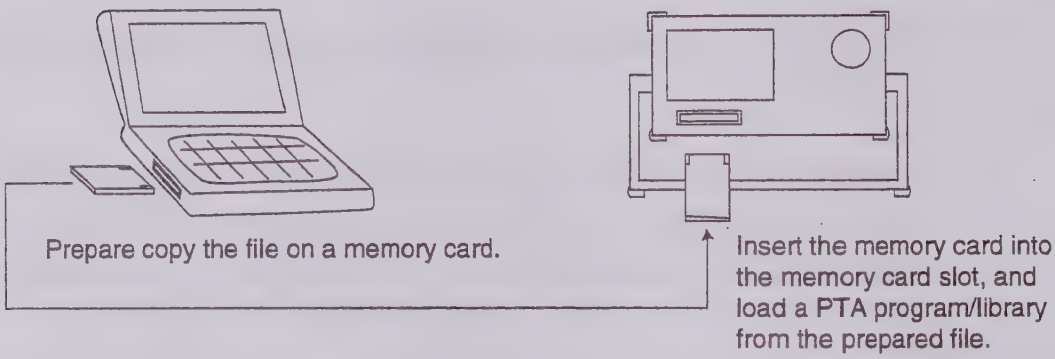
A PTA library is loaded and executed on menus following the [SHIFT] + [PTA]Å®[PTA Library : F2] keys. The PTA library can be executed by registering it to a menu of the [User] key and pressing the appropriate Fkey.
Also, the PTA library can be executed by directly inputting the PTA library name as a remote control command from the controller.



A PTA program or PTA library can be loaded to the execution memory of MS2665C/67C/68C by either of the following three methods:

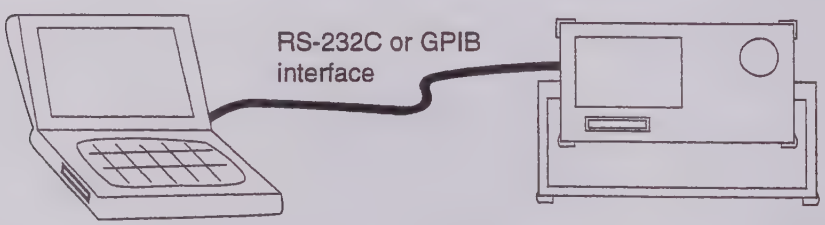
(1) Prepare a PTA program/library as a text file of in DOS format on a memory card, and load it to spectrum analyzer.

- Prepare the PTA program/library file using the edit program (editor) on the personal computer.
- Copy the prepared file to the memory card.
- Insert the memory card to the memory card slot of spectrum analyzer, and load it from the operation menu of the PTA program or PTA library.


Prepare copy the file on a memory card.
Insert the memory card into the memory card slot, and load a PTA program/library from the prepared file.

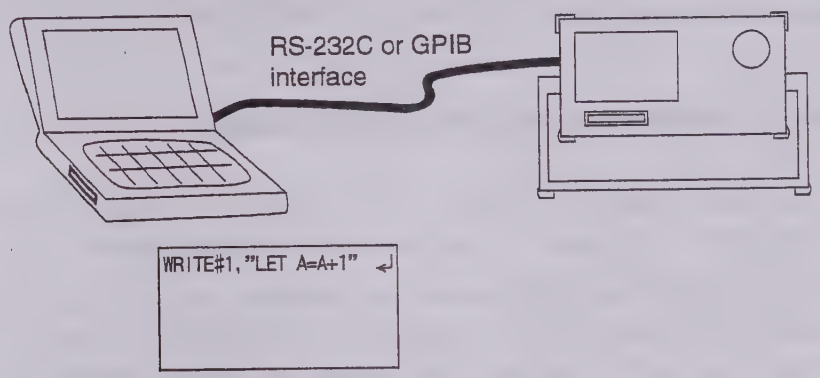
(2) Prepare a PTA program/library file on the personal computer, and load it to spectrum analyzer via the RS-232C or GPIB interface.

- Prepare the PTA program/library file using the edit program (editor) on the personal computer.
- Load the data (PTL statement) of the prepared file to spectrum analyzer via the RS-232C or GPIB interface.



(3) Remote-controlling spectrum analyzer from the personal computer, directly input the PTL statement.

- Remote-control spectrum analyzer from the personal computer via the RS-232C or GPIB interface and get the PTA operation screen.
- Sending a PTA statement line by line to spectrum analyzer, prepare a PTA program/library on the execution memory of spectrum analyzer.



Operations Related to PTA Program

Operations related to the loading and execution of PTA programs are described below. Operations are the same as those of the PTA functions and PTA program execution provided in the existing measuring instruments of our make (for example, MS2601B, MS2602A, MS8604A, etc.).

Startup of PTA

PTA is actuated by pressing the [SHIFT] + [PTA : 7] keys on the front panel or inputting the remote control command "PTA_1". The screen is erased and the cursor appears at the home position (top left of the screen).

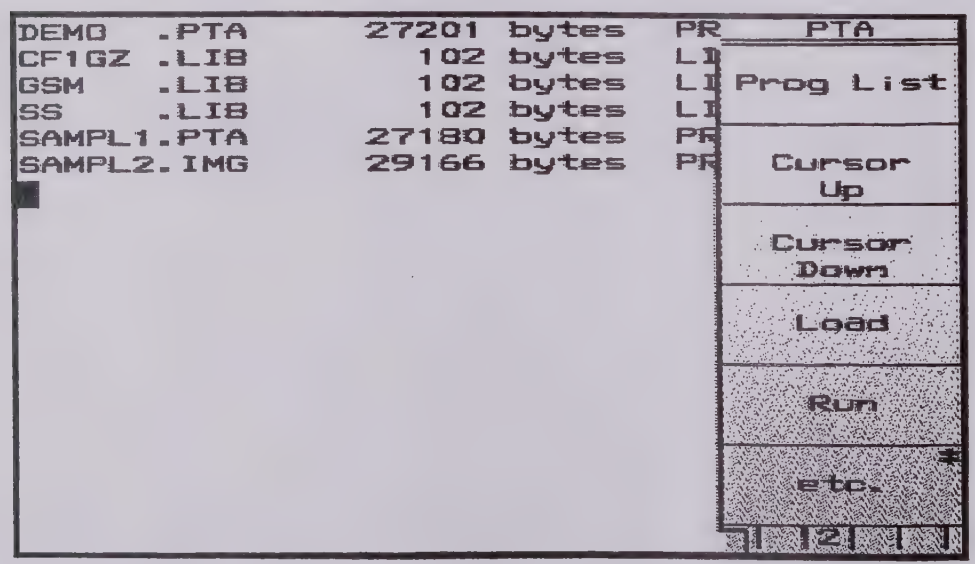
Additionally, by registering a PTA program/library as a startup program, it can be actuated and executed upon powering on. (For details about the startup registration of the PTA program, see Section 3 "STARTUP command". Likewise, for details about the PTA library, see Section 3 "POWERUP command".)

Loading the PTA program from memory card

A PTA program can be prepared as a text file in DOS format on a memory card and loaded to spectrum analyzer by the edit program (editor) of the personal computer and the like.

(1) Press [SHIFT] + [PTA : 7] → [PTA Program : F1] keys and get the PTA program operation mode (PTA ON).

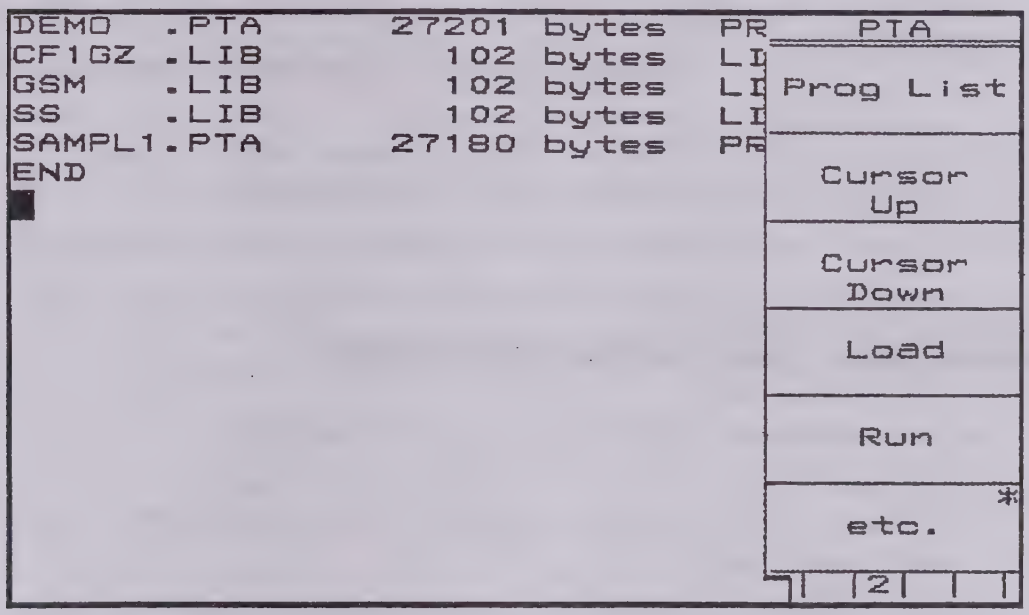
(2) Press the [PLIST : F1] key of the PTA program menu (page 2) to display a list of program names stored in the memory card.



(3) Press [CURSOR UP : F2] and [CURSOR DOWN : F3] keys and move the cursor to the program name to load.

(4) Press the [LOAD : F4] key.

Read out the PTA program from the memory card. When reading is completed, the [END] message is displayed.



(5) Press the [RUN : F5] key to execute the program.

(6) To stop execution, press the [RESET : F4] key of the PTA program menu (page 1).

Execution, stop of the PTA program

After loading a PTA program from a memory card, the PTA program can be executed and stopped without loading operation. Since the execution memory of the PTA program is backed up by batteries, it is retained under the loaded condition after powered off. Condition under execution is not retained.

(1) Press [SHIFT] + [PTA : 7]→[PTA Program : F1] keys and get the PTA program operation mode (PTA ON).

(2) Press the [RUN : F1] key of the PTA menu (page 1) to execute the program.

(3) To interrupt program execution, press the [STOP : F2] key.

(4) To resume program execution, press the [CONT : F3] key.

(5) To stop program execution, press the [RESET : F4] key. To restart execution, press the [RUN : F1] key.

PTA termination

To terminate PTA, press the [RESET : F4] key to stop program execution, and then press the [PTA OFF : F5] key or input a remote control command "PTA␣0".

Afterwards, the screen (which has been displayed by display subroutine) is cleared to be returned to ordinary measurement screen.

Note

For the display subroutine, see Section 5, "System Subroutines".

Format of PTA program file

There are two formats for a PTA program file on a memory card, as follows:

(1) Text format

The extender for a PTA program file in text format is ".PTA". An example of the PTA program file in text format is shown below.

```
10 '=========================================================
20 '== MS2660 series PTA Program/Library Sample Program   ===
30 '=========================================================
40 '
50 HOME&ERASE'                          Erase PTA screen
60 PRINT "   Hello PTA World!!"'        Print message
70 PUT   "IP"'                          Preset MS2660 series
80 PUT   "CF 100MHZ"'                   Set center frequency 100MHz
90 PUT   "SP 100KHZ"'                   Set frequency span 100kHz
100 PUT  "MKPK"'                        Perform peak search
110 STOP'                               Stop execution
```

(2) Execution format

The extender of a PTA program file in execution format is ".IMG". The PTA program file in execution format is stored in the form of binary data and cannot be edited on the personal computer.

The file in execution format can be prepared by adding ".IMG" as the extender to the file name by the LOAD command of PTA. Storing the file in execution format will reduce loading time.

Operations Related to PTA Library

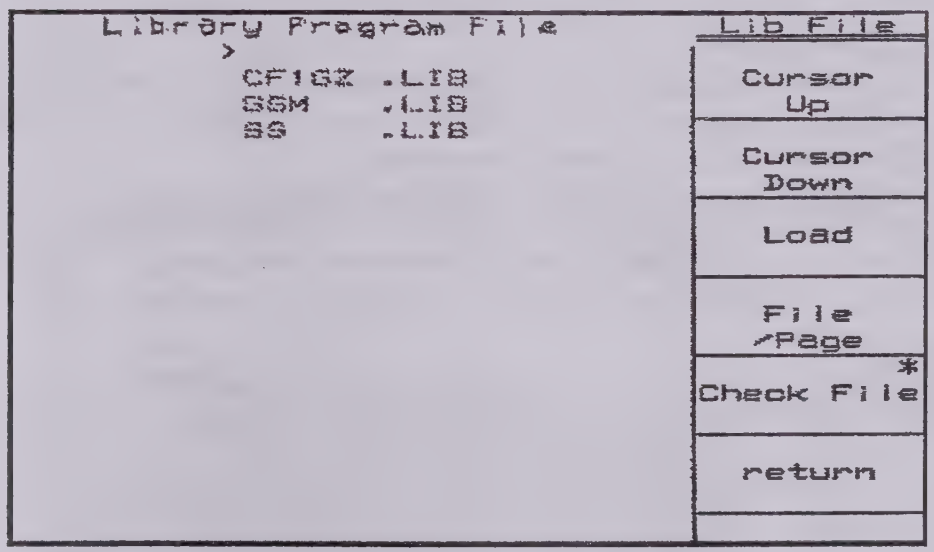
Operations related to the loading and execution of the PTA library are described below.

Loading the PTA library from memory card

A PTA library can be prepared as a text file in DOS format on a memory card and loaded to spectrum analyzer by the edit program (editor) of the personal computer and the like.

(1) Press [SHIFT] + [PTA : 7] → [PTA Library : F2] keys and get the PTA library operation mode (PTA ON).

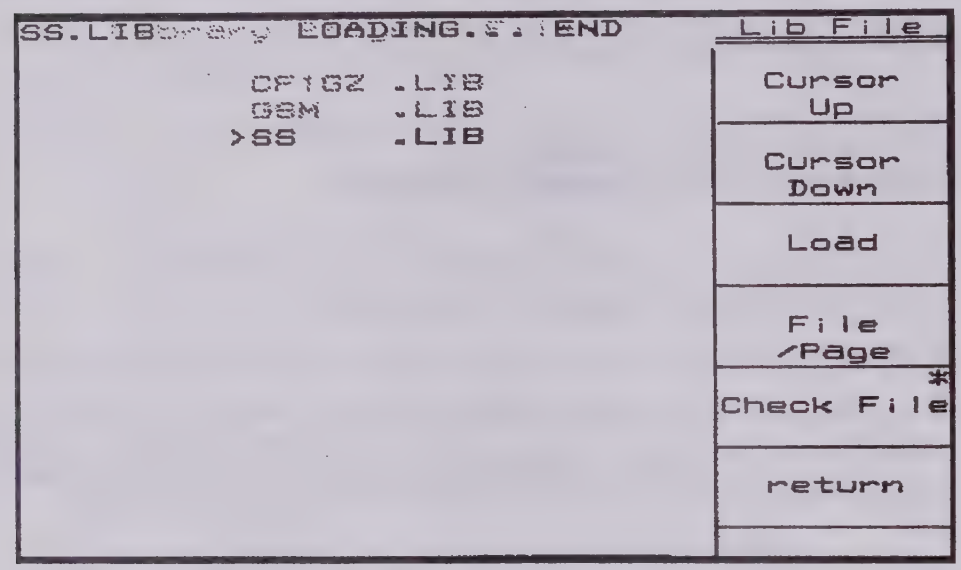
(2) Press the [Library File : F2] key of the PTA library menu to display a list of library files stored in the memory card. If the list cannot be displayed at a time, press the [File/Page : F4] key to display the next page.



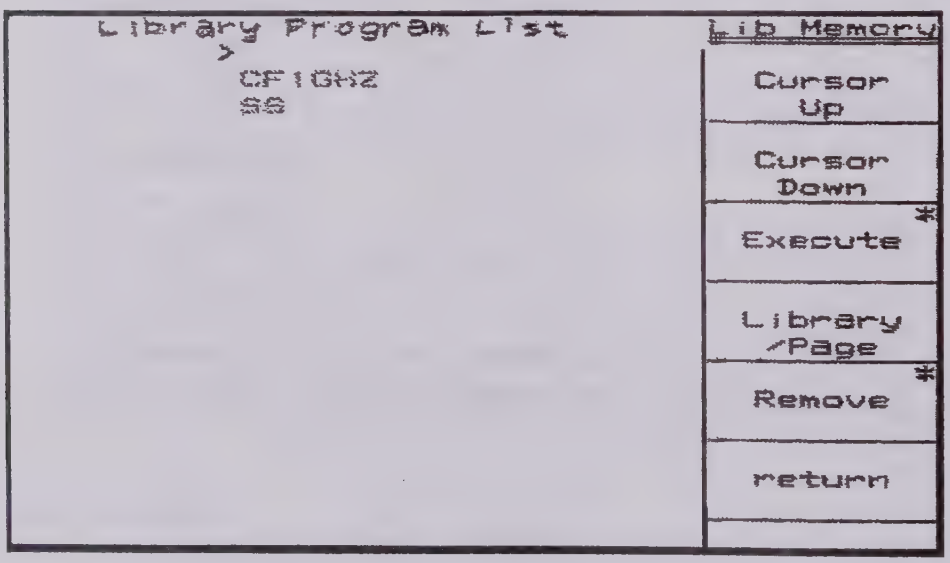
(3) Press [CURSOR UP : F1] and [CURSOR DOWN : F2] keys and move the cursor to the library file name to load.

(4) Press the [LOAD : F3] key.

Read out the PTA library from the memory card. When reading is completed, the [LOADING...END] message is displayed.



After loading, the PTA library loaded on the execution memory can be displayed in list form by pressing the [Library Memory : F1] key of the PTA library menu.

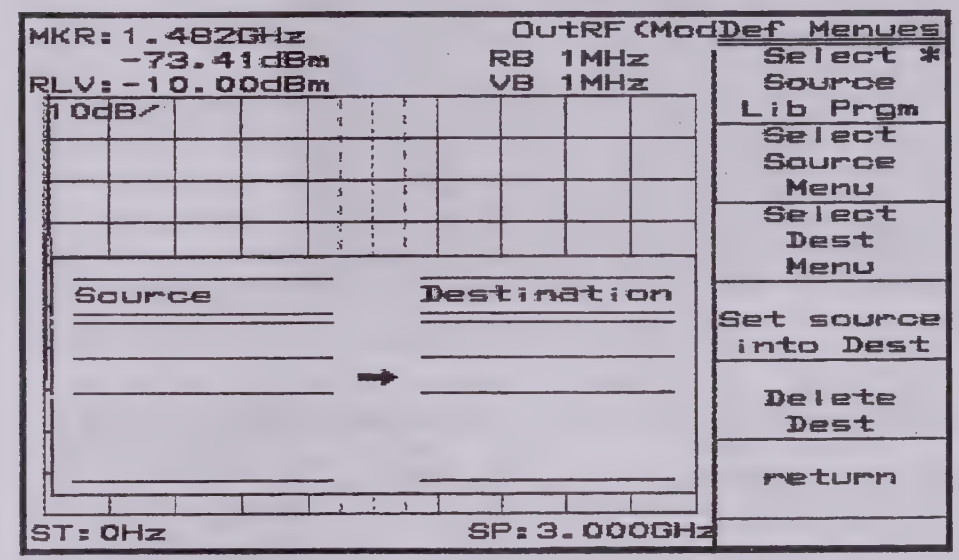


Also test execution can be done by operating menus following the [Executed : F3] key.

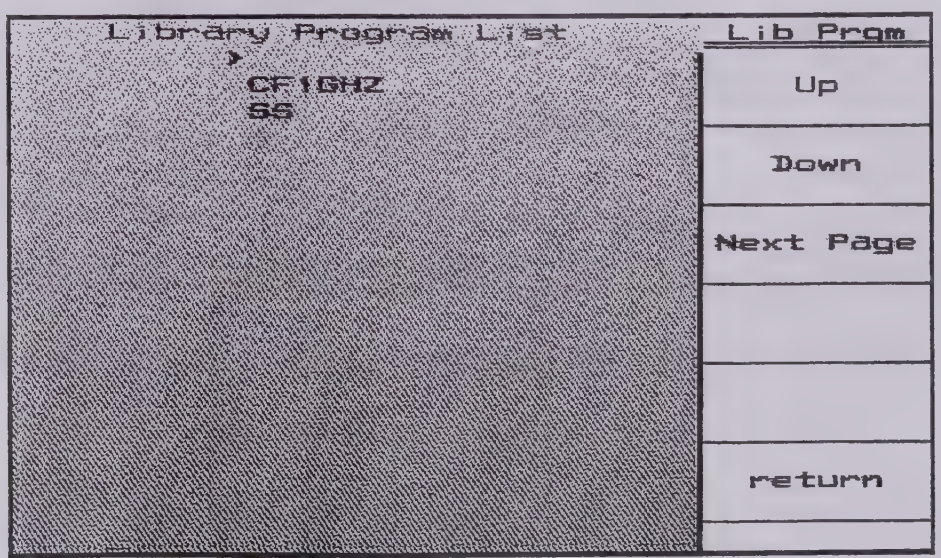
Registering the PTA library to user key

The PTA library loaded to the execution memory can be executed by registering it to a menu of the [User] key on the front panel. The registering operation procedure of the PTA library to the User key described below.

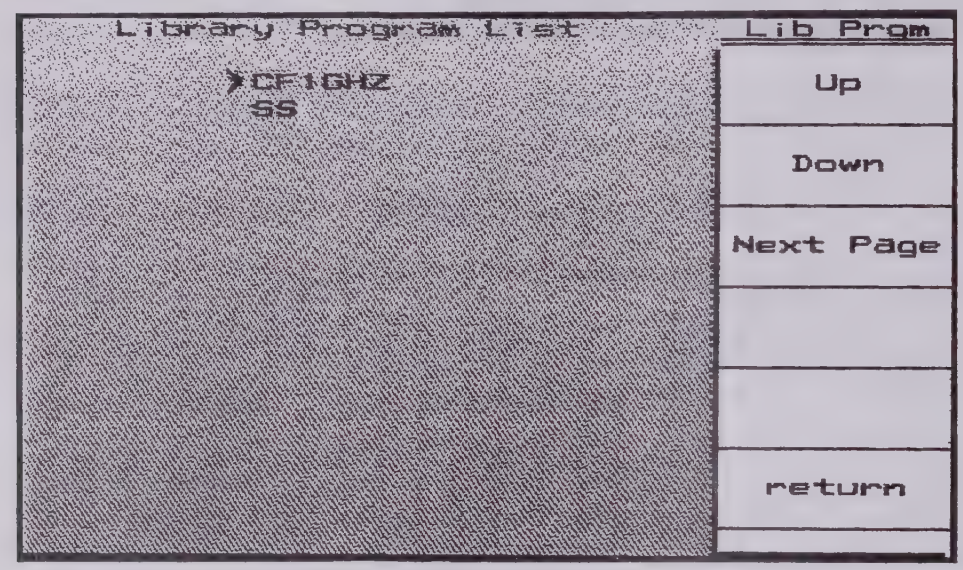
(1) Press [SHIFT] + [User Define : 8] → [Define Menus : F1] keys and display the User key registration screen.



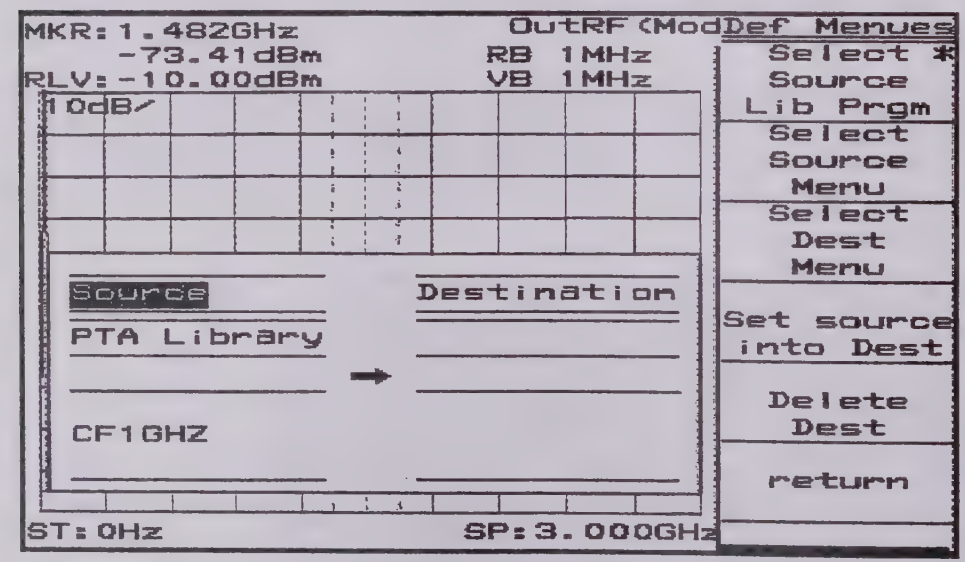
(2) Press the [Select Source Lib prgm : F1] key. The PTA library loaded in the execution memory is displayed in list form.



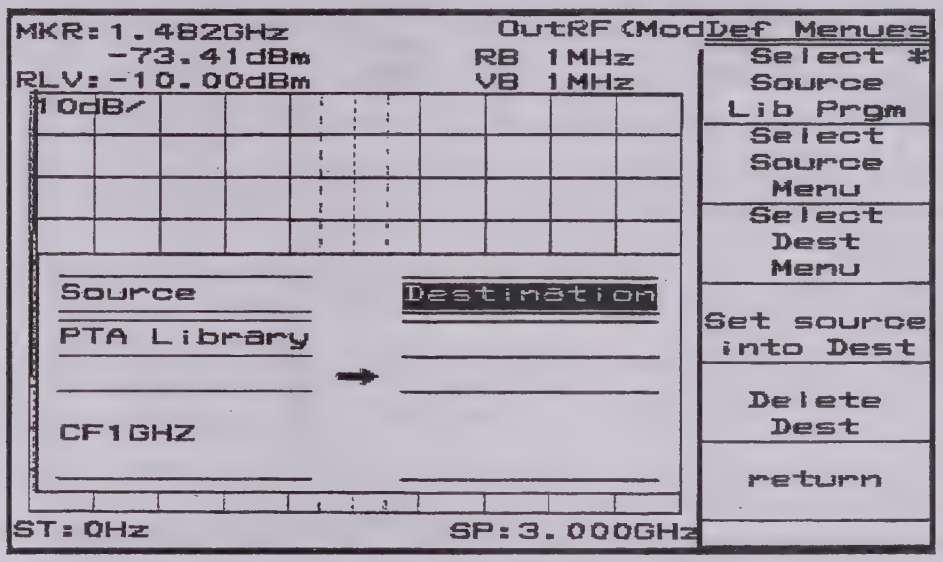
(3) Press [UP : F1] and [DOWN : F2] keys, and move the cursor to the library name to load to the User key.



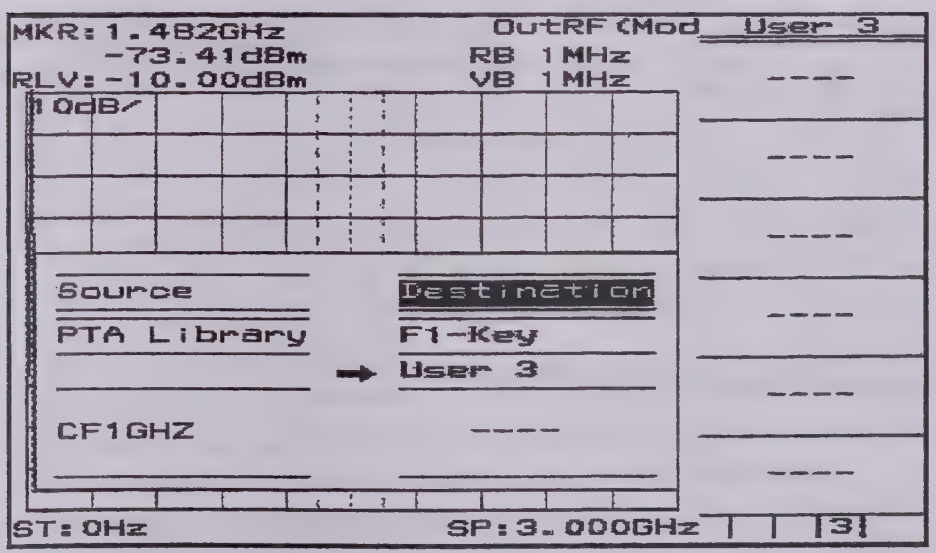
(4) Press the [return : F6] key. The selected library name is displayed in the Source column of the User key registration screen.



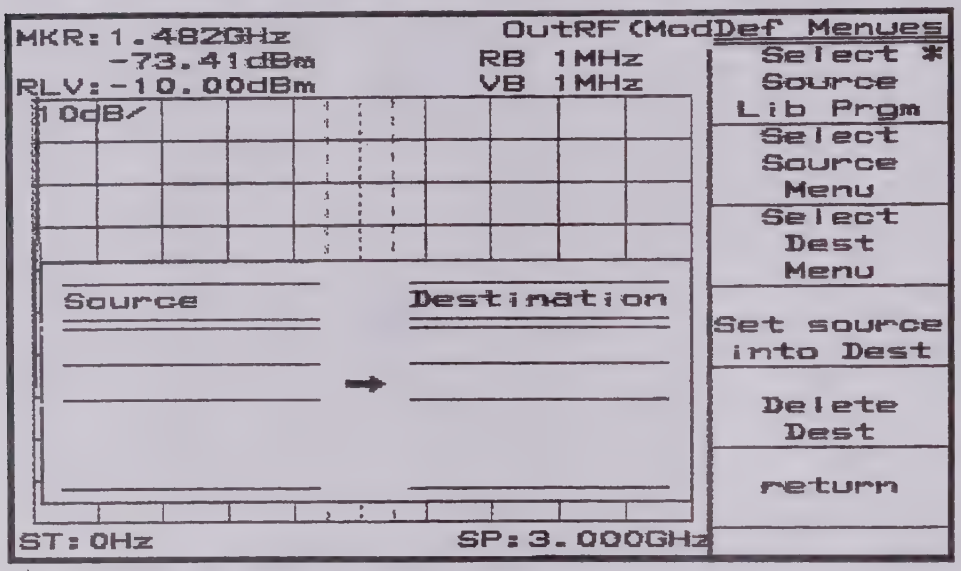
(5) Press the [Select Dest Menu : F3] key. the title in the Destination column of the User key registration screen is inverted, indicating the waiting status for the selection of the destination menu.



(6) Press the [User] key on the front panel and press a menu to register. Each time a menu is pressed, the selected menu is displayed on the Destination column of the User key registration screen. A menu that is pressed last is the destination.

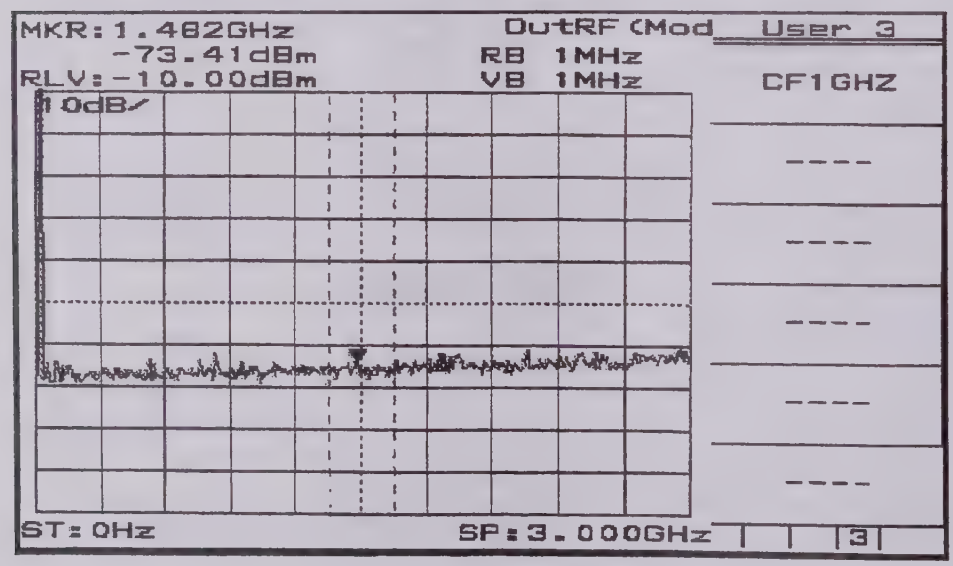


(7) Press [SHIFT] + [User Define : 8] → [Define Menus : F1]→[Set source into Dest : F4] keys to register the execution of the PTA library to the selected User key.



After registering, pressing the [return : F6] key erases the User key registration screen.

Press the [User] key on the front panel and look at the registered menu; the PTA library name is displayed on the menu, indicating that registration is completed.



Press this key to start executing the registered PTA library.

Execution, stop of the PTA library

The PTA library loaded to the execution memory is normally executed by registering it to the User key, but test execution can be done from the PTA library menu.

(1) Press [SHIFT] + [PTA : 7] → [PTA Library : F2] keys and get the PTA library operation mode.

(2) Press the [Library Memory : F1] key and display the PTA library loaded on the execution memory in list form. If the list cannot be displayed at a time, press the [File/Page : F4] key to display the next page.

(3) Press [CURSOR UP : F1] and [CURSOR DOWN : F2] keys and move the cursor to the program name to test-execute.

(4) Press the [Execute : F3] key and get the PTA library test execution mode.

Under the test execution mode, the following operations are available:

(5) Press the [RUN : F1] key to execute the library.

(6) To interrupt library execution, press the [STOP : F2] key.

(7) To resume library execution, press the [CONT : F3] key.

(8) To stop library execution, press the [RESET : F4] key. To restart execution, press the [RUN : F1] key.

Format of PTA library file

There are two formats for a PTA library file on a memory card, as follows:

(1) Text format

The extender for a PTA library file in text format is ".LIA". One PTA library file in text format can store one PTA library only. The title of this PTA library is the same as that of the PTA library file. Data in the PTA library file in text form is totally the same as that of the PTA program, with only an exception of the extender of the file.

(2) Execution format

The extender of a PTA library file in execution format is ".LIB". The PTA program file in execution format is stored as binary data and cannot be edited on the personal computer.

One PTA library file in execution format can store plural PTA libraries. There are no title relations between the PTA library file and PTA libraries stored in it.

Operations related to PTA library

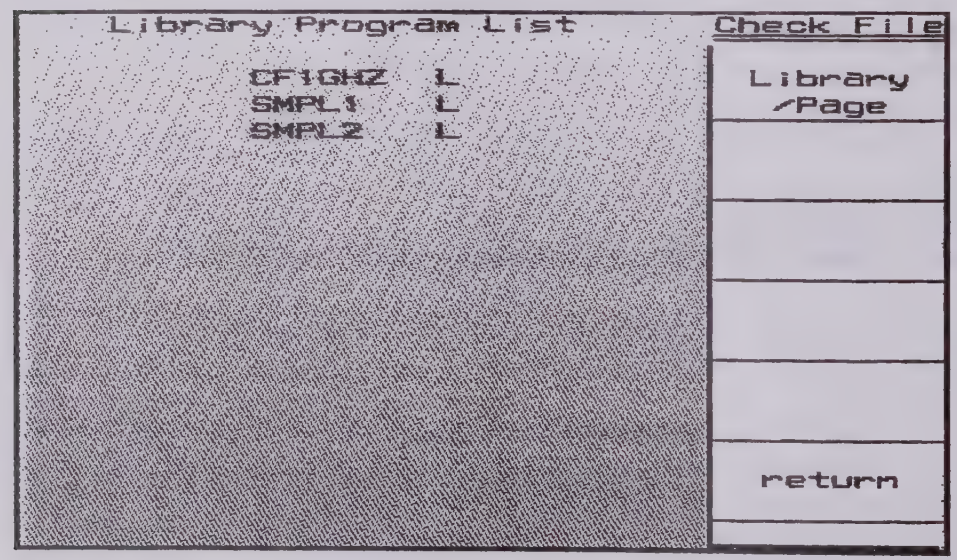
In the case of a PTA library file in execution format, stored PTA libraries cannot be confirmed by a file list. For this purpose, the PTA libraries can be listed by the following operations:

(1) Press [SHIFT] + [PTA : 7] → [PTA Library : F2] keys and get the PTA library operation mode.

(2) Press the [Library File : F2] key of the PTA library menu to display a list of library files stored in the memory card. If the list cannot be displayed at a time, press the [File/Page : F4] key to display the next page.

(3) Press [CURSOR UP : F1] and [CURSOR DOWN : F2] keys and move the cursor to the library file name to confirm PTA libraries stored in it.

(4) Press the [Check File : F5] key.
A list of PTA library files stored in the selected PTA library file is displayed on the screen. If the list cannot be displayed at a time, press the [File/Page : F1] key to display the next page.



Panel Key Operations during PTA Program/Library Execution

Data input keys

The soft keys, numeric keys, and unit keys on the front panel serve as data input keys.

(1) F1, F2, F3, F4 and F5 keys

The F1 to F5 keys are referred to in the program and correspond to the system variables EX1, EX2, EX3, EX4 and EX5 respectively.

Each time the key is pressed, the variable contents are alternately changed to 0 or 1. All the data in these variables are 0 at initial state and resetting. Displayed name in menu can be defined with DEF subroutine.

Note

For EX1, EX2, EX3, EX4 and EX5, see Section 5, " System Variables".

(2) YES and NO keys

These are typing aids for the INPUT statement; the "YES" and "NO" character string can be input by a single key operation.

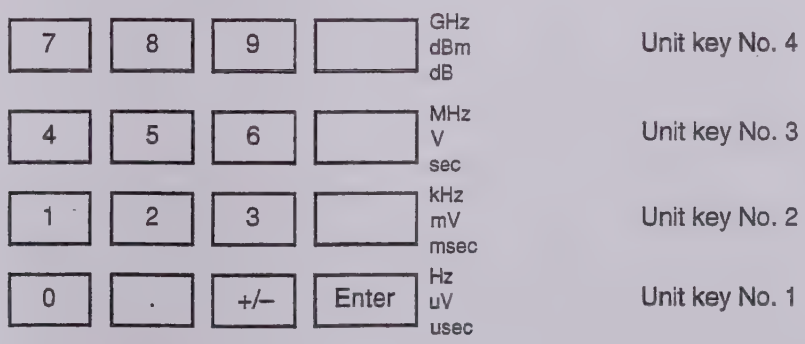
(3) Numeric keys

These are the [0] to [9] ,[.] and [BS] keys which are used for inputting data on INPUT statement. Press the [Enter] key to terminate the input; use the [BS] key to delete one character.

(4) Unit keys

Unit key No. 1 : Treats this key as the CR key.
Unit key No. 2 : Treats this key as the [,] key.
Unit key No. 3 : Treats this key as the [–] key.
Unit key No. 4 : Invalid

*: The figure below shows unit key numbers.



Operation of other panel keys

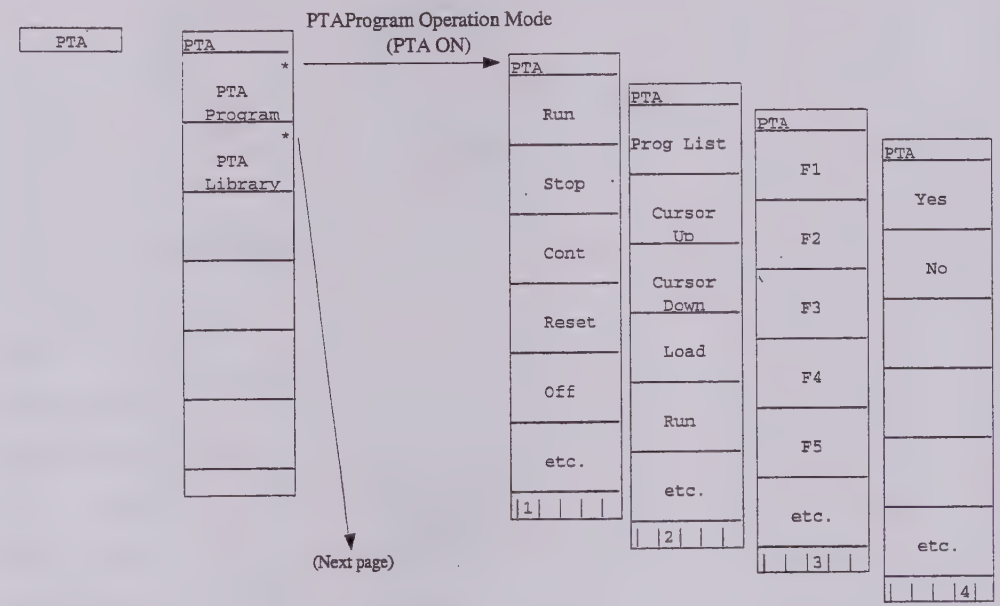
When PTA is ON, the panel keys are locked-out except for the number/[Enter] keys, [Shift] key, [Local] key and soft keys (F1 to F6).

Menu Construction of the PTA Key

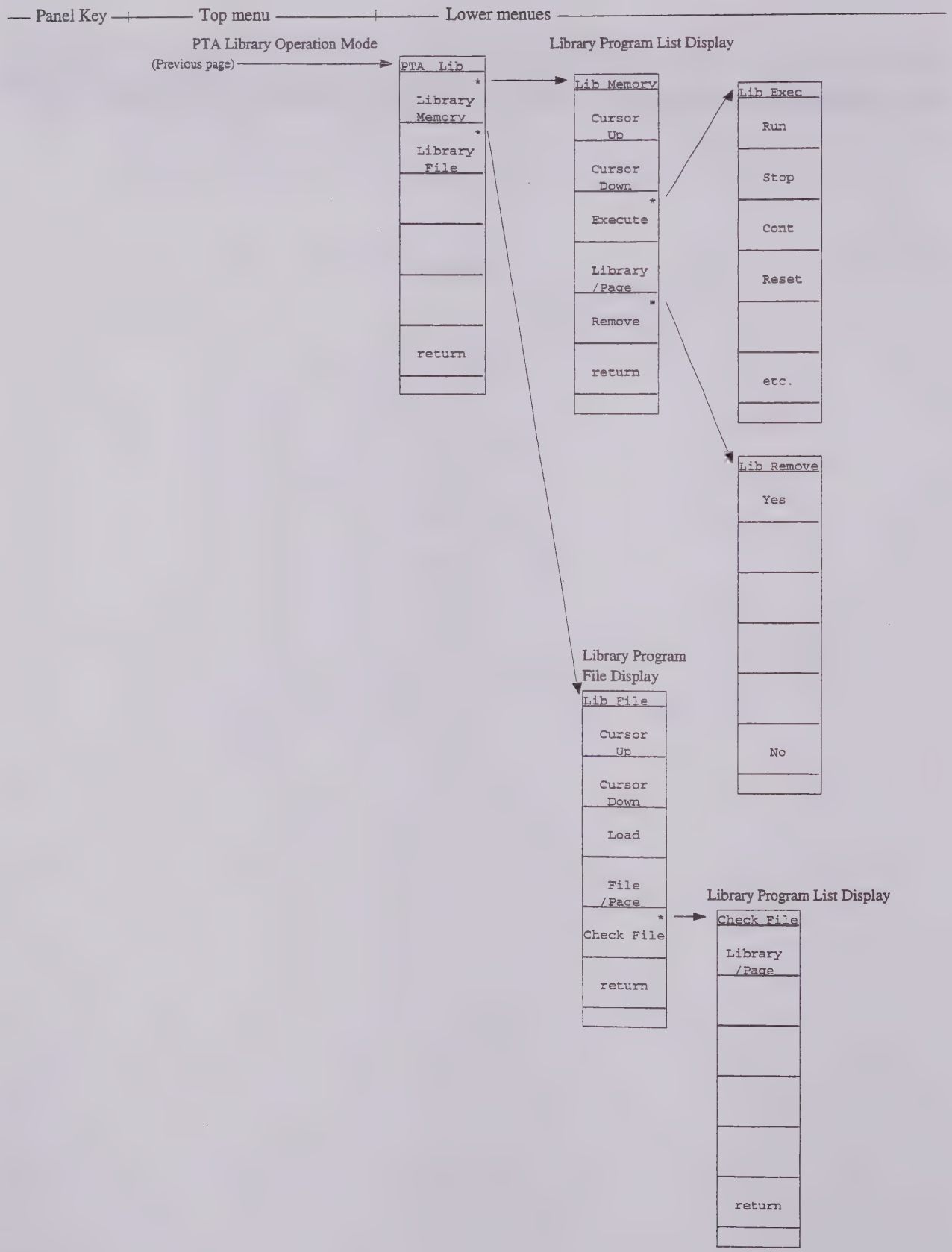
Menu layers following [SHIFT] + [PTA : 7] keys are shown below.

Menu Tree

Panel Key — Top menu — Lower menues



Menu Tree


Panel Key
Top menu
Lower menues
PTA Library Operation Mode
(Previous page)
PTA Lib
Library Memory
Library File
return
Library Program List Display
Lib Memory
Cursor Up
Cursor Down
Execute
Library /Page
Remove
return
Lib Exec
Run
Stop
Cont
Reset
etc.
Lib Remove
Yes
No
Library Program File Display
Lib File
Cursor Up
Cursor Down
Load
File /Page
Check File
return
Library Program List Display
Check File
Library /Page
return

SECTION 3

PTL COMMANDS

TABLE OF CONTENTS

Program Input Command 3-4

PCOPY Command 3-5

DELETE Command 3-6

RENUM Command 3-7

LIST Command 3-8

LISTG Command 3-9

PMEMO Command 3-10

Immediate Execution Command 3-11

RUN Command 3-12

STOP Command 3-13

CONT Command 3-14

RESET Command 3-15

SAVE Command 3-16

LOAD Command 3-17

OVERLAY Command 3-18

PDEL Command 3-19

PLIST Command 3-20

STARTP Command 3-21

CANCEL Command 3-22

EDITLIB Command 3-23

EDITPTA Command 3-24

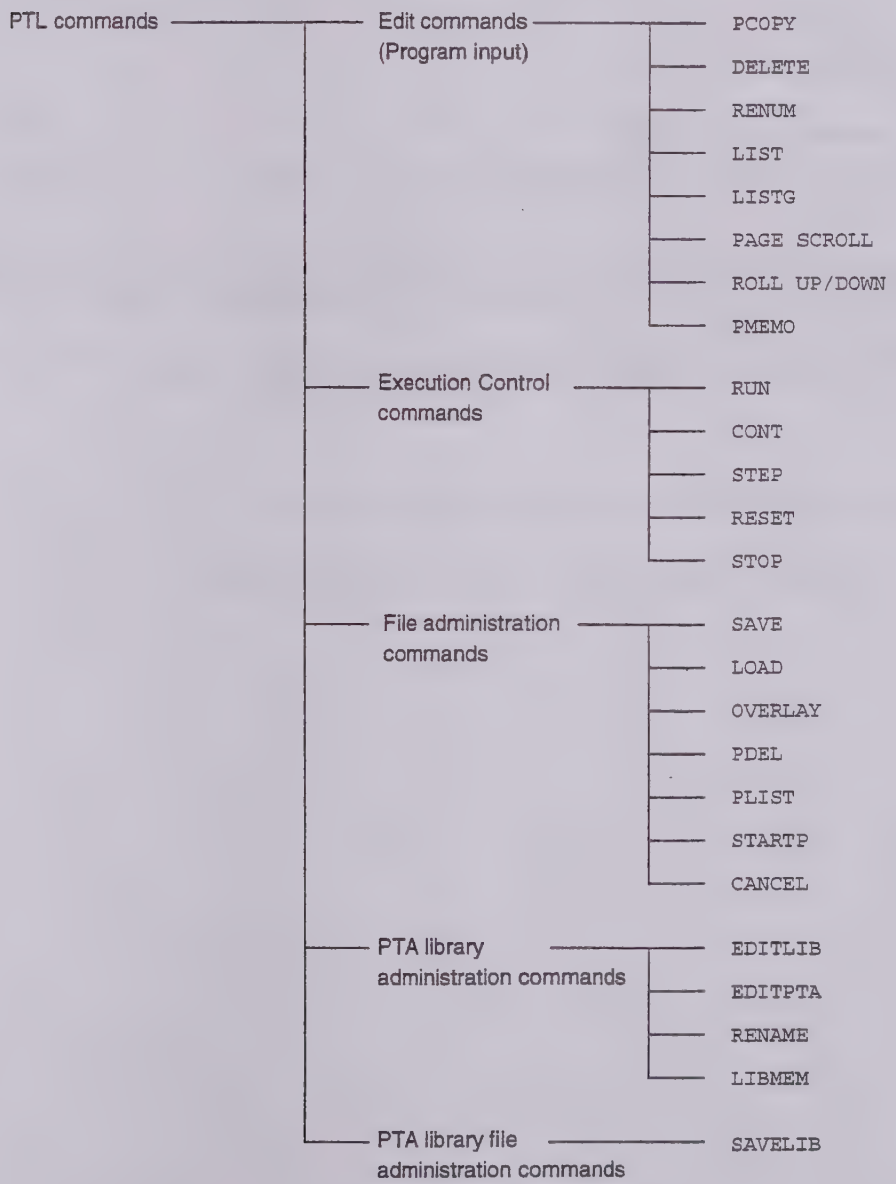
RENAME Command 3-25

LIBMEM Command 3-26

SAVELIB Command 3-27

SECTION 3
PTL COMMANDS

PTL (Personal Test Language) commands include commands for the edition, execution and filing of the PTA programs/libraries, and are composed as shown below:



Program Input Command

(1) Function

When a statement with a line No. is inputted, it is stored as a PTA program/library to the program area. When the line No. is different from those already inputted, the statement is added or inserted, and when the line No. is the same, the statement will replace the already inputted statement.

(2) Format

```
Line number Statement
```

Integer constant from 1 to 65535

Notes:

- When 111 or more characters (including the line number) are input on one line during program input, the program on that line may not be displayed during LIST-command execution after execution of the RENUM command.
- For a description of the RENUM command, see Section 3, "RENUM Command".

PCOPY Command

(1) Function

This statement copies the specified program.
(from <copy-source start-line number> to the <copy-source end-line number>) in the unit of increment specified by <increment> from the <new start-line number>.
If <increment> is omitted, thenÅe10Åfis used as the default value.

(2) Format

```
PCOPY operand 1,[operand 2],operand 3,operand 4
```

New start-line number Increment Copy-source start-line number Copy-source end-line number.

1) `PCOPY 1ØØ,,1Ø,3Ø` Copies the statement (from lines 10 to 30) to location 100 in increments of 10 and labels all sequent.

2) `PCOPY 1ØØ,5,1Ø,3Ø` Copies the statement (from lines 10 to 30) to location 100 and in increment of 5 and labels all sequent.

Notes:

- If the line number of a newly-copied statement is identical to the line number of the current statement, ERROR F101 occurs.
- If a line has more than 111 characters when PCOPY is executed, display is disabled during LIST command execution.

DELETE Command

(1) Function

This command deletes all or part of a program.

(2) Format

```
DELETE [operand 1][,][operand 2]
       operand 1≤operand 2
```

(3) Example

1)	`DELETE`	Deletes entire program and initializes variable values.
2)	`DELETE 1ØØ`	Deletes statement on line 100.
3)	`DELETE 1ØØ,`	Deletes statements on lines 100 to the end line.
4)	`DELETE ,5ØØ`	Deletes statements on start line to line 500.
5)	`DELETE 1ØØ,5ØØ`	Deletes statements on line 100 to line 500.

- When deleting only a line, it is possible by Line number [RETURN].

RENUM Command

(1) Function

This command renumbers line numbers used in the program. When the increment value or new line number is omitted, 10 is used as the default value.

(2) Format

```
RENUM [[operand 1][,][operand 2][,][operand 3][,][operand 4]]
```

New-line number | Increment | Start-line number | End-line number

1) `RENUM` Renumbers all program statements starting from first-line number 10 in increments of 10

2) `RENUM 1ØØ` Renumbers all program statements from new first-line number 100 in increments of 10

3) `RENUM 1ØØ,5` Renumbers all program statements starting from new first-line number 100 in increments of 5

4) `RENUM 1ØØ,,5Ø` Renumbers statements (from line number 50 through the last line) starting from new first-line number 100, in increments of 10

5) `RENUM 1ØØ,5,5Ø` Renumbers statements (from line number 50 through the last line) starting from new first-line number 100, in increments of 5

6) `RENUM 1ØØ,,,15Ø` Renumbers statements (from line number 10 through 150) starting from new first-line number 100 in increments of 10

7) `RENUM 1ØØ,,5Ø,15Ø` Renumbers statements (from line number 50 through 150) starting from new first-line number 100 in increments of 10

8) `RENUM 1ØØ,5,,15Ø` Renumbers statements (from line number 10 through 150) starting from new first-line number 100 in increments of 5

9) `RENUM 1ØØ,5,5Ø,15Ø` Renumbers statements (from line number 50 through 150) starting from new first-line number 100 in increments of 5

Notes:

- Labels can be used for operands 1, 3 and 4.
- "ERROR F101" occurs if there is a line number larger than that of operand 4 when operand 1 is smaller than operand 4.
- If the number of characters on a line is more than 111 characters, when the number of lines of the program line becomes two lines or more with RENUM command, ERROR F20 will occur during LIST command execution and display the lines.

LIST Command

(1) Function

This command displays all or part of a program on the LCD.

(2) Format

```
LIST [operand 1][,][operand 2]
       |                |
  Start-line number   End-line number
       operand 1≤operand 2
```

1) `LIST` Lists entire program
2) `LIST 1ØØ` Lists the statement on line 100
3) `LIST 1ØØ,` Lists statements on line 100 to end line
4) `LIST ,5ØØ` Lists statements on start line to line 500
5) `LIST 1ØØ,5ØØ` Lists statements on line 100 to 500

Note: Labels can be used for operands 1 and 2.

LISTG Command

(1) Function

This command outputs all or part of a program to a printer connected to the RS-232C/GPIB/parallel (centronics) interface.

(2) Format

```
LISTG address [[,][operand 1][,][operand 2]]
      |
```

Address of printer (0 to 30)

Operand 1 and operand 2 in the LISTG command are used in the same way as the LIST command.

Notes:

- To use RS-232C/GPIB/parallel (centronics) interface from PTA, it is necessary to choose a port to use. The selection of the port, press [SHIFT] + [Interface : .] keys, and then press the [Connect to Peripheral : F6] key several times.
- When the program is output to the RS-232C or parallel (centronics) interface, addresses have no meaning, but they should be specified as a formality.

PMEMO Command

(1) Function

This command displays on the screen the used memory size of the program area in which a PTA program/ library is stored and the memory size required to store to a memory card.

(2) Format

```
PMEMO
```

(3) Output example

```
Used memory size:          262 bytes
 PTA program               262 bytes
 LIB programs                0 bytes
 Variables                   0 bytes
Unused memory size:      196295 bytes

File size:
 PTA program  (ASCII)        161 bytes

              (BINARY)       334 bytes
 LIB programs (BINARY)        72 bytes
```

Total size of used memories of program area

Not used

Memory size required to store to memory card

Immediate Execution Command

(1) Function

When a statement with no line number is input and the ↵ (RETURN) key is pressed, the statement is immediately executed.

However, GOTO, GOSUB, RETURN, RETMAIN, IF, FOR, NEXT DATA, RDATA, RESTORE and CHAIN, CALLIB statements are not immediate execution commands.
See Section 4 for these statements.

(2) Format

```
Statement
```

RUN Command

(1) Function

This command starts PTA program/library execution. Execution is terminated when the STOP statement is executed, when an error occurred, or when the [RESET] key is pressed.

(2) Format

```
[RUN] key or
RUN [[operand 1][,operand 2]]
```

Start-line number Suspended-line number

1) `RUN` Starts execution from statement on first line
2) `RUN 1ØØ` Starts execution from statement on line 100
3) `RUN ,5ØØ` Starts execution from statement on first line, and suspends execution on line 500
4) `RUN 1ØØ,5ØØ` Starts execution from statement on line 100, and suspends execution on line 500

Note: Contents of variables are not initialized by the RUN command.

STOP Command

(1) Function

This command stops the PTA program/library in execution.

(2) Format

```
[STOP] key
```

CONT Command

(1) Function

This command resumes the suspended program execution.

Note that this command can only be executed when program execution is suspended after execution of the RUN or STEP command.

(2) Format

```
[CONT] key
CONT [operand]
```

1) `CONT` — Restarts program from next on suspended line.
2) `CONT 1ØØØ` — Restarts program from next on suspended line, and suspends execution on line 1000.

RESET Command

(1) Function

This command stops command or PTA program/libraries execution.

(2) Format

```
[RESET] key
```

(3) Initialization

This Command : 1. Clears system variables EX1, EX2, EX3, EX4, and EX5.
2. Clears user-defined variables. Common variables are not cleared.

SAVE Command

(1) Function

This command saves a PTA program to a memory card. In this case, the file size of the PTA program must be smaller than the unused memory size of the memory card.
The file size of the PTA program and the unused memory size of the memory card are output on the screen by executing the PMEMO command and the PLIST command, respectively.

(2) Format

```
SAVE PTA program name [.Attribute][,operand 1][,operand 2]
```

- PTA program name: Alphanumeric string up to 6 characters starting with an uppercase alphabetic character.
- .Attribute: .PTA or .IMG
- operand 1: Start-line number
- operand 2: End-line number

Notes:

- The file opened by CALL OPNI (or OPNO) "% file name" is closed when this command is executed.
- Labels can be used as operands 1 and 2.
- Before saving a program, make sure the memory card is formatted. When saving to an unused memory card, format the memory card in advance.
  For formatting mathod of the memory card, refer to paragraph 4.5.2 of Panel Operation Part in the Operation Manual.
- When .PTA is specified as attribute, the program is saved as an ASCII file. When .IMG is specified, the program is saved as a binary file, which has a shorter loading time. As the default attribute, .PTA is automatically selected for saving.

LOAD Command

(1) Function

This command loads a PTA program loaded on a memory card and stores it to the program area in the main frame. All the PTA programs already stored in the user program area are replaced by the new program unless OVERLY is executed.

(2) Format

```
LOAD PTA program name [.Attribute]
```

Alphanumeric string up to 6 characters starting with an uppercase alphabetic character.

.PTA or .IMG

Notes:

- The file (opened by CALL OPNI (or OPNO) "% file name") is closed when this command is executed.
- When reset during program loading, part of the programs is loaded.
- The spectrum analyzer program area (memory) is backed up by a battery. Therefore, the program contents are not lost even when the power switch is turned off.

OVERLAY Command

(1) Function

This command specifies to overwrite the current PTA program during LOAD command execution.

(2) Format

```
OVERLAY
```

Note: This state continues until the RESET command is executed.

PDEL Command

(1) Function

This command deletes the PTA programs stored in a memory card.

(2) Format

```
PDEL PTA program name or PTA library file name [.Attribute]
                                                    ↑
                                          PTA, IMG, LIB, LIA
```

Notes:

- "% file name" (data files) cannot be erased by the PDEL command.
- The file (opened by CALL OPNI (or OPNO) "% file name") is closed when this command is executed.
- When attribute is omitted, .PTA is automatically selected as the default attribute for saving.

PLIST Command

(1) Function

This command displays on the CRT screen the names and sizes of files stored on memory card along with the amount of unused memory.

(2) Format

```
[PLIST] key
```

(3) Output

This command causes the screen to scroll by page (24 lines) unit.
When more than 17 files are stored on a memory card, the files cannot be displayed on one page, therefore a screen such as 1) below is displayed. The screen is displayed page by page by using the PLIST command repeatedly. When the contents can be displayed on a single page, a screen such as 2) is displayed.

1) When pages follow

```
.....          .....bytes     PROG (IMAGE)
%SDAT0.DAT     1024 bytes     DATA
%SDAT2.DAT     1024 bytes     DATA
ABCXYZ.PTA     15808 bytes    PRJG (ASCII)
                                        continue
```

2) When no pages follow

```
BANDLH.PTA     18568 bytes    PROG (ASCII)
RPLLH.IMG      35786 bytes    PRJG (IMAGE)
MAXMIN.LIB     27368 bytes    LIBRARY
unused memory size : 89010 bytes
```

Unused memory size : Indicates unused memory size (No. of bytes) of the memory card.

NOTES

- The file (opened by CALL OPNI (or OPNO) "% file name") is closed when this command is executed.
- Only the PTA program file, PTA library file and data file created by the PTA are displayed by the PLIST command. Therefore, since the spectrum analyzer does not display the saved waveform and measurement parameters, if they exist, the unused memory size is reduced.

STARTP Command

(1) Function

Turns on the PTA and registers the start-up function, which loads and executes the specified PTA program when the power is turned on.
This function can be separately registered and set for a PTA program on a memory card and a PTA program in the main frame.

(2) Format

`STARTP program name` : Register for PTA program on memory card
`STARTP @` : Register for `spectrum analyzer` internal PTA program

1) Start-up function registration for PTA program on memory card
   - When the power is turned on after this function is registered, the PTA is turned on and the registered PTA program is loaded and executed.
   - When this function is registered, a special "p2110. bat" file is created on the memory card. (This file is not displayed by the PLIST command.)
   - In the following cases, the start-up function is not performed even if registered:
     - When a memory card is not inserted when the power is turned on.
     - When a PTA program with the registered program name is not found on the memory card.
     - If the power was turned on while pressing the [PTA : 7] key.
   - This function is executed first even if start-up function is registered for the internal program of the main frame.
   - When start-up function is executed, the PTA program is loaded from the memory card, and the previous program in the main frame is cleared. Also, when start-up function is registered for the internal PTA program, it is cleared too.
   - If both "STARTP" and "STARTP@" are registered, the file registered by the STARTP command is executed preferentially.
2) Start-up function registration for spectrum analyzer internal PTA program
   - When the power is turned on after this function is registered, the PTA is turned on and the spectrum analyzer battery back-up PTA program is run automatically.
   - When there is no PTA program in the spectrum analyzer, this function cannot be registered.
   - The start-up function is not performed in the following cases:
     - When the memory card start-up function was executed first.
     - When a new PTA program was loaded after the start-up function was registered. (In this case, start-up function registration is canceled.)
     - When there is no PTA program in the spectrum analyzer.
     - If the power was turned on while pressing the [PTA : 7] key.

CANCEL Command

(1) Function

Cancels start-up function registration.

(2) Format

`CANCEL` : Register for PTA program on memory card

`CANCEL @` : Cancel registration for `spectrum analyzer` internal PTA program

- When start-up registration for memory card is canceled, the "p2110. bat" file is deleted.
- When the power is turned on while pressing the [PTA : 7] key, the start-up function is temporarily canceled, but the function registration status does not change.

EDITLIB Command

(1) Function

This command defines a new PTA library, or specifies a PTA library as the object of the program execution and program edition commands.

(2) Format

```
EDITLIB [PTA library name]
```

Alphanumeric string with up to 8 characters starting with a capital alphabet

Characters available for the 2nd character on :

- Under bar
- Capital alphabet : A to Z
- Small alphabet : a to z
- Numeral : 0 to 9

However, small alphabets are converted to capitals.

- When the EDITLIB command is executed specifying the name of a new PTA library as a parameter, the registration of the specified PTA library is started. The PTA library can be registered by inputting a statement with a line No.
- When the EDITLIB command is executed specifying the name of an already registered PTA library as a parameter, a library program to be the object of program execution and edition commands is specified.
- When the EDITLIB command is executed without a parameter, the name of the currently specified library is displayed.
- The PTA library name specified by the EDITLIB command is displayed at the bottom right of the screen.

EDITPTA Command

(1) Function

This command specifies PTA programs as the object of edition and execution.

(2) Format

```
EDITPTA
```

- Select PTA programs as the object of edition and execution. The object of processing is switched to PTA programs by executing the EDITPTA command during PTA library selection. Additionally, immediately after PTA ON, always PTA programs are selected.

RENAME Command

(1) Function

This commands changes the name of the specified PTA library.

(2) Format

```
RENAME PTA library name,PTA library name
```

New PTA library name

Program name to be changed
Alphanumeric string with up to 8 characters starting with a capital alphabet
Characters available for the 2nd character on :
- Under bar
- Capital alphabet : A to Z
- Small alphabet : a to z
- Numeral : 0 to 9

However, small alphabets are converted to capitals.

- The name of an already registered PTA library is changed. It is not allowed to specify the already registered PTA library name for the new PTA library name.

LIBMEM Command

(1) Function

This command displays a list of PTA libraries in the memory.

(2) Format

```
LIBMEM
```

- Names of library programs in the memory are displayed in list form. If the list cannot be displayed at a time, re-execute the LIBMEM command to display the next page. If there is no library in the memory, nothing is displayed.

SAVELIB Command

(1) Function

This command saves the specified measuring instrument library program to a memory card with the specified file name.

(2) Format

```
SAVELIB File name [,PTA library name...]
```

Alphanumeric string with up to 8 characters starting with a capital alphabet

Characters available for the 2nd character on :
- Under bar
- Capital alphabet : A to Z
- Small alphabet : a to z
- Numeral : 0 to 9

However, small alphabets are converted to capitals. Library names can be specified up to ten names by separating them with commas (,). If no name is specified, all the PTA libraries residing in the memory are specified.

Alphanumeric string with up to 6 characters starting with a capital alphabet Characters available for the 2nd character on :
- Under bar
- Capital alphabet : A to Z
- Small alphabet : a to z
- Numeral : 0 to 9

However, small alphabets are converted to capitals.

- The PTA library is saved in intermediate code form. The file extender is ".LIB".

SECTION 4

PTL

TABLE OF CONTENTS

Elements of Statement Configuration 4-3
- Line number 4-3
- Constants 4-4
- Variables 4-6
- Multi statement 4-8
- Functions 4-9
- Arithmetic operators 4-14
- Relational operators 4-15
- String concatenation (the "+" operator) 4-16
- Formats 4-17
- Label 4-18

Basic Statements 4-19
- Comment (REM statement) 4-19
- Array declaration (DIM statement) 4-20
- Initialization (CLEAR statement) 4-22
- Substitution (LET statement) 4-23
- Branch (GOTO statement) 4-24
- Termination of execution (STOP statement) 4-24
- Branch to subroutines (GOSUB statement) 4-24
- Return from subroutines to main routine (RETMAIN statement) 4-25
- Return from subroutines (RETURN statement) 4-25
- Decision (IF statement) 4-26
- Repetitions start (FOR statement) 4-27
- Repetition termination (NEXT statement) 4-28
- Key-input (INPUT statement) 4-29
- Display (PRINT statement) 4-30
- Reverse display (PRINTR statement) 4-34
- Positioning the cursor (LOCATE statement) 4-35
- Data statement (DATA statement) 4-35
- Reading data (RDATA statement) 4-36

Read specification of data statement (RESTORE statement) 4-36
Setting measurement parameters (PUT and WRITE 1000 statements) 4-37
Measurement parameter/data read
(GET, COM and READ 1000 statements) 4-38
Program loading and execution (CHAIN statement) 4-40
ENABLE EVENT statement 4-40
DISABLE EVENT statement 4-44
ON EVENT statement 4-44
RETINT statement 4-45
Character size specification (DCHSIZE statement) 4-46
Home position (HOME statement) 4-47
Delete (ERASE statement) 4-47
Time wait (WAIT statement) 4-47
System subroutine execution (CALL statement) 4-48
ON ERROR statement 4-48
OFF ERROR statement 4-49
RETERR statement 4-49
RETRY statement 4-50
RESUME statement 4-50
GIVEUP statement 4-51
Error branch (ERROR statement) 4-51
Error main (ERRMAIN statement) 4-52
Data input 1 (READ statement) 4-52
Data input 2 (BREAD statement) 4-53
Data input 3 (WREAD statement) 4-53
Data output 1 (WRITE statement) 4-54
Data output 2 (BWRITE statement) 4-54
Data output 3 (WWRITE statement) 4-55
Data writing to the dual port memory (WDPM statement) 4-57
Data reading from the dual port memory (RDPM statement) 4-57
S.O.S (SOS) 4-58
Setting the pseudorandom number sequence (RNDMIZE statement) 4-59
Calling the PTA library (CALLIB statement) 4-59
Removing the PTA library from program memory (REMOVE statement) 4-60
Clearing common variables (COMCLEAR statement) 4-61
Setting CALLIB parameter values (PARASET statement) 4-61
Loading the PTA library file LOADLIB statement) 4-62

SECTION 4
PTL

PTL (Personal Test Language) is a programming language similar to BASIC.
It consists of basic PTL statements and extended PTL (including system variables, system subroutines, and GPIB statements).

Elements of Statement Configuration

Line number

(1) Function

A line number is placed at the beginning of each statement and serves as an index during program editing or execution.

(2) Format

```
Numeric String
   |
```

Integer constant from 1 to 65535

Constants

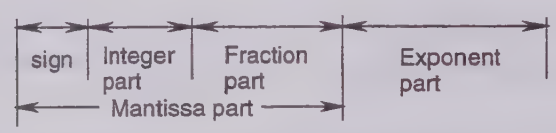
(1) Function

A constant represents a specific numeric value, character string or bit string.

(2) Format

(a) Numeric constants

```
[-]numeric string[.numeric string][E[-]numeric string]
```



The maximum number of mantissa digits is 15(including a sign and a decimal point.) and the range of exponent part is 10^{308} to10^{-307} .
When a numeri c constant is assigned to an integer type numeric variable, the range is –32768 to +32767.

(b) Character constants

```
"String"
```

1 to 255 characters enclosed with double quotation marks (" ")

Note: One line of program corresponds two lines on screen. Then, maximum number of characters on a program line is limited to the value.

(c) Bit constants

- Hexadecimal constant
  $ Hexadecimal expression
  0 to FF
- Binary constant
  # Binary expression
  0 to 11111111

(3) Examples

(a) Numeric constants

```
1
-12.3
12E3                ...Equal to 12000
-Ø.12E-3            ...Equal to  -0.00012
```

(b) Character constant

```
"Who are you? "
```

(c) Bit constants

```
$F                  ...Equal to #1111 (binary) or 15 (decimal).
#ØØØ11Ø1Ø           ...Equal to $1A (hexadecimal) or 26 (decimal)
```

Variables

Variables include local, common and system variables. For the system variable, see Section 5, "System Variables".

(1) Local variables

A local variable is one that is effective in a PTA program/library only.
Local variables include simple and array variables.

- Simple variable

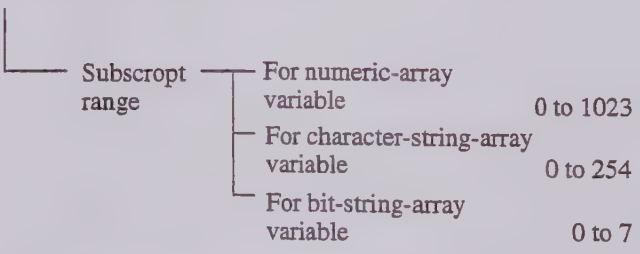
There are numeric, character string, and bit string variables. The simple variable consists of eight or less characters, the first of which must be an upper-case alphanumeric character as shown below:

- Real number-type numeric-variable name: Upper-case alphabetic character [alphanumeric [alphanumeric]] ——— ABCD0123
- Integer-type numeric-variable name: Upper-case alphabetic character [alphanumeric [alphanumeric]] % ——— A%
- Character-string-variable name: Upper-case alphabetic character [alphanumeric [alphanumeric]] $ ——— ABC$
- Bit-string-variable name: Upper-case alphabetic character [alphanumeric [alphanumeric]] # ——— A#

- Array variable

The variable (declared as an array by the DIM statement) is called an array variable. Some system variables are also handled as array variables. The format of the array variables is shown below.

- Array variable : variable (numaric constant or numeric variable)



NOTES

- The subscript range for an array variable is from 0 to array size –1.
- When the subscript in the array variable is a real number, it is truncated after the decimal point.
- Up to 256 variables can be used (except for system variables).
- Pre-registered symbols (such as commands, statements, functions and system variables) cannot be used as user-defined variable names.

(2) Common variables

Common variables are ones that can be commonly accessed from all programs (PTA program/library). The name of a common variable starts with "@" followed by capital alphabets. The length of a common variable name is 8 characters at longest, including the @ mark.
Values of common variables are retained until the RESET command or COMCLEAR command is executed.
Common variables include simple variables and array variables:

- Simple variables

There are numeric, character string and bit string variables.

- Real number variable name: @ + variable name
- Integer numeric variable name: @ + variable name + %
- Character string variable name: @ + variable name + $
- Bit string variable name: @ + variable name + #

- Array variables

Like array local variables, array common variables are declared by a DIM statement.

The DIM statement may be declared in any of programs, and double definition is also allowed. The array size is linear or quadratic.

- Real number variable name: @ + variable name (array size [, array size])
- Integer numeric variable name: @ + variable name + % (array size [, array size])
- Character string variable name: @[alphanumerics[alphanumerics]]$ (array size [, array size])
- Bit string variable name: @ @[alphanumerics[alphanumerics]]# (array size [, array size])

Multi statement

By using ' & ' as the delimiter in a statement, multiple statements can be entered on the same line.
This delimiter can also be used to enter a program of two lines. There are no restrictions on the number of statements within a program, provided that the length of the program does not exceed two lines.

Example :
```
1Ø FOR I=Ø TO 1Ø & A=I*I & PRINT A & NEXT I
2Ø STOP
```

Functions

There are basic functions (arithmetic, boolean, statistical and character-string functions) and dedicated functions in PTL. The system functions are used for measurement evaluation.

(1) Arithmetic function

Function name	Function	Parameter	
Sine	SIN(X)	The X unit is degrees.	A constant or a variable os used for X.
Cosine	COS(X)		
Tangent	TAN(X)	X≠±90(2n+1), n:any integer	
Arcsine	ASN(X)	|X| ≦1	
Arccosine	ACS(X)		
Arctangent	ATN(X)		
Natural logarithm	LN(X)	X>0	
Common logarithm	LOG(X)		
Exponent	EXP(X)		
Square root	SQR(X)	X≧0	
Absolute value	ABS(X)		
Sign	SGN(X)	FOR X>0 , SGN(X)=1 FOR X<0 , SGN(X)=–1 FOR X=0 , SGN(X)=0	
Integer value	INT(X)	X : Numeric type constant variable (An integer less than X is returned.)	
Rounding up	ROUND(X[,N])	X : Numeric type constant variable N : Numeric type constant variable (default value: N=0) (X is rounded up to the N-th decimal place.)	
Function to calculate the quotient and remainder	Q=DIV(R,S,D)	Q : Numeric variable ------ Stores the quotient R : Numeric variable ------ Stores the remainder S : Numeric variable ------ Stores the dividend D : Numeric variable ------ Stores the divisor	
Function to isolate the integer and decimal parts of a real number	I=FIX(S,D)	I : Integer variable ------- Stores only the integer part S : Real-number variable --- Stores the real number of the original value D : Real-number variable --- Stores only the decimal part	

(2) Boolean functions

Function name	Function	Parameter
Negation	NOT(X)	X and Y are constants and variable of bit type or numeric type, and hexadecimal constants.
Logical product	AND(X,Y)	
Logical sum	OR(X,Y)	
Exclusive OR	EOR(X,Y)	

(3) Statistical functions

Function name	Function	Parameter
Function to find maximum value	MX=max(S)	S : Variable defined as one-dimensional arrey MX : Stores the maximum value MN : Stores the minimum value SM : Stores the sum total MS : Stores the mean value VR : Stores the variance $\text{Variance} = \frac{\Sigma (X-\bar{X})^2}{\text{No of samples}}$
Function to find minimum value	MN=min(S)	
Function to find sum	SM=sum(S)	
Function to find mean value	MS=mean(S)	
Function to find variance value	VR=var(S)	
Function to find all above values	VR=sta (S,MX, MN,SM,MS)	

NOTES

The left side always consists of numeric variable in which found (calculated) value is stored.
The one-dimensional S-parameter is valid even if there is only one element provided.
When all the elements are to be processed statistically, no subscript is necessary at the entry.
If a subscript is included, only the element specified by the subscript will be processed.

(4) Character-string functions

(a) Interchange between numerics and characters (strings)

1. ASC (Alphabetic constant or variable)
   ASC generates the character code for the first character of the string.

2. CHR$ (Constant or variable)
   CHR$ generates the character with the character code corresponding to the parameter value.
   For a character type, the character remains unchanged. The parameter range is from 0 to 255.

3. STRING$ (Numeric constant or variable, constant or variable, character constant or variable)
   STRING$ generates the characters (with the character code of the numeric value or the first character of string specified by the 2nd parameter) by the number of characters specified by the 1st parameters.Up to 255 repetitions may be specified.
   Refer to CHR$ ( )

4. HEX$ (numeric-value-type constant or variable 1 [, numeric-value-type constant or variable 2])
   A decimal value of the first parameter is given as a hexadecimal character string with number of digits specified by the 2nd parameter.
   An error will occur if the value of the first parameter does not fall in between -2^{31} and $2^{32}-1$. An error will occur if the second parameter goes beyond eight digits. When omitted, the return value will be of variable length.

5. OCT$ (Constant or variable)
   OCT$ generates the octal character string corresponding to the parameter value. An error is generated when the range –32768 to 32767 is exceeded.

6. BIN$ (numeric-value-type constant or variable I [, numeric-value-type constant or variable 2])
   A decimal value of the first parameter is given as a binary character string with number of digits specified by the 2nd parameter.
   An error will occur if the value of the first parameter does not fall in between -2^{31} and $2^{32}-1$. An error will occur if the second parameter goes beyond 32 digits. When omitted, the return value will be of variable length.

7. CVI (Character constant or variable of 2 or more characters)
   CVI generates the value converted from a character string to an integer numeric expression. If the character string exceeds two characters, the excess part is disregarded. Conversely, an error is generated when it is less than 2 characters.

8. CVD (Character constant or variable of 8 or more characters)
   CVD generates the value converted from a character string to a double-precision real-number numeric expression. When the character string exceeds 8 characters, the excess part is disregarded. Conversely, an error is generated when it is less than 8 characters.

9. MKI$ (Integer constant or variable)
   MKI$ generates the corresponding character code of the internal binary expression of the specified numeric value. This is the reverse process of the previously-mentioned CVI.

10. MKD$ (Double-precision real-number constant or variable)

MKD$ generates the corresponding character code of the internal binary expression of the specified numeric value. This is the reverse process of the previously-mentioned CVD.

11. VAL (Character variable, Number constant or variable 1, numeric constant or variable 2)

VAL isolates the mth to nth numeric characters (including other than numeric code) of the specified data string and changes them to the double-precision real-number numeric expression, assuming that m and n are the specified values by variable 1 and variable 2, respectively.

Both m and n may be omitted. When m is omitted, the object runs from the head character of the data string: and when n is omitted, the object runs to the last character of the data string.

An error occurs when no numeric character is found.

12. BVAL (character constant or variable)

This function will convert the parameter string notated in binary into an unsigned decimal value.

An error will occur if the parameter exceeds 32 bits. All characters other than "0" or "1" will be ignored.

13. HVAL (character constant or variable)

This function will convert the parameter string notated in hexadecimal into an unsigned decimal value.

An error will occur if the parameter exceeds 32 bits (8 characters). Characters other than "0" to "9" and "A" to "F" are ignored.

14. CHR (Numeric constant or variable)

CHR generates the same character string as that to be displayed by the PRINT statement within the specified numeric value by parameter.

15. STR$ (Numeric constant or variable)

This performs exactly the same processing as described for the CHR function.

(b) Retrieving character strings

1. INSTR ([Numeric constant or variable,] character constant or variable 1, character constant or variable 2)

When character string 2 is found within character string 1, its position is returned; if it is not found, 0 is returned. When the numeric value is included in the 1st parameter, the search starts from the indicated position with the numeric value; when it is omitted, the search starts from the header. The range of the value is from 1 to 255.

2. LEFT$ (Character constant or variable, numeric constant or variable)

This gives the specified number of characters (counting from the left) as specified by the second-parameter. When the specified number exceeds the number of characters in the strings, whole the character string is given. The specifiable number is from 0 to 225. When the specified number is 0 , a null string is returned.

3. MID$ (Character constant or variable, numeric constant or variable 1, numeric constant or variable 2)

This gives the n of character strings from the m-th character, assuming that the m and n are the specified values by the variable 1 and variable 2, respectively. The range of m/n is (1 to 256) / (1 to 255), respectively. When m exceeds the total number of characters, a null string is returned.

4. RIGHT$ (Character constant or variable, numeric constant or variable)
   This performs the same processing as the LEFT$ ( ) command but from the right side. The value range is also the same (0 to 255). Note that this command does not reverse the character string sequence.
5. LEN (Character constant or variable)
   LEN gives the number of characters in a character string including all character codes from 0 to $1F.
6. SLEN (character type constant or variable)
   This gives the number of characters composing a character string in the same manner as specifying a value in LEN ( ). However, this gives the length with the space at the end of the character string omitted .
7. SGET$ (character type constant or variable)
   This gives a valid character string with the space at the end omitted.

(5) Dedicated functions

Function description	Function	Parameter
Reads the error code and line number in which error occurred on	`V=ERRREAD(m)`	m 0 : Error code 1 : Line number in which error occurred
Reads the type of event	`A#=STATUS(m)`	m 0 : Event 0 1 : Event 1 2 : Event 2 3 : Event 3
Reads the date and o'clock, minute, second	`A$=DTREAD$(m)`	m 0 : Date (YY-MM-DD) 1 : o'clock, minute, second (HH:MM:SS)
Random number generation (more than 0, less than 1)	`RND(m)`	m : Specify an arbitrary value.

NOTES

- ERRREAD (m) can only be used during at error interrupt. For details on error interrupts, see Section 4, "ON ERROR statement".
- STATUS (m) can only be used during an event interrupt. For details on event interrupts, see Section 4, "ENABLE EVENT statement".
- m is a numeric constant or numeric variable.
- The sequence of pseudo-random numbers generated by RND(m) becomes the same each time RUN is executed.
  See Section 4, "RNDMIZE statement" for how to change the sequence.

Arithmetic operators

(1) Function

These operators perform addition, subtraction, multiplication, division, and exponential operations.

(2) Format

= ... Substitution
+ ... Addition
- ... Subtraction
* ... Multiplication
/ ... Division
! ... Exponentiation
( ) ... Represents operation priority
(Operations in parentheses are performed first.)

(3) Operation Priority

The operation priority is shown below.

Table 4-1 Operation priority of arithmetic operators

Operation priority	Arithmetic operators
High ↕ Low	!
	* /
	+ -
	=

NOTES

- Bits and characters cannot be used in operations.
- If X of X ! Y is a minus number, but Y is a plus number, X ! Y can be operated.
- If there is a different type variable on the right side of an equals sign (=), an overflow or underflow error may occur.
- Number of digits of divided becomes number of digits of the solution on division with numerals or variables.

(4) Example

```
A$="abc"
C=(D+1ØØ)/E
J=((K+1)*1Ø-M)*1Ø
```

Relational operators

(1) Function

These operators perform relational operations.

(2) Format

=	...	Equal (=)
>< or <>	...	Not equal (≠)
>	...	Greater than (>)
<= or =<	...	Equal to or less than (≤)
<	...	Less than (<)
>= or =>	...	Equal to or greater than (≥)

(3) Comparing character strings

When comparing the sizes of character strings, count only significant characters.
(Ignore any spaces at the ends of the character strings to the left and right of an operator)

- If two character strings are the same length, their characters are compared sequentially from the beginning. The first character which is different is found. The character which has the lower code value will determine the smaller character string.

  Example : ABC is smaller than ABX.

- If two character strings are different lengths, the character strings over their common length are compared. If the two strings are equal over this length, the shorter character string will be the smaller character string.

  Examples : ABX is larger than ABCD.
  ABC is smaller than ABCD.

- The smallest character string is one with 0 length.

  Example : The length of A$ is 0 when DIM A# (10) is declared.

(4) Examples

```
IF C=Ø GOTO 1ØØ

IF JKL>=168 STOP
```

String concatenation (the "+" operator)

(1) Function

String concatenation is possible with the "+" operator.

(2) Format

```
{ character string constant  }     { character string constant  }
{ character string variable  } + { character string variable  }
{ character string function  }     { character string function  }
```

Notes:

- Only be used with the right hand parameter of the LET statement.
- You cannot concatenate character string and numeric values, character string and bit, or bit and bit.

(3) Examples

```
100  A$="ABC"
110  B$="DEF
120  A=INSTR(A$,"␣")-1
130  B=INSTR(B$,"␣")-1
140  C$=LEFT$(A$,A)+LEFT$(B$,B)
150  PRINT "A$=",A$
160  PRINT "B$=",B$
170  PRINT "C$=",C$

AS=ABC␣␣␣␣␣␣␣␣
B$=DEF␣␣␣␣␣␣␣␣
C$=ABCDEF␣␣␣␣␣
                └── Space
```

NOTES

- Simple character-string variables are assumed to be a ten-character array-declared variables, implicitly. Therefore, characters not assigned will be filled with spaces. For details, see Section 4, "Display (PRINT statement)" and "Reverse display (PRINTR statement)".
- By using the above method, you can concatenate actual stored character only.

Formats

(1) Function

These formats specify the format of strings in output operations. Integers, real numbers without exponents, real number with exponents, strings, binary numbers, and hexadecimal numbers can be specified.

(2) Formats

- Integer
  : I number of digits
  (1 to 18)
- Real number without exponent
  : F number of all digits. number of fractional digits
  (4 to 20)
  (Number of all digits ≥ number of fractional digits+3)
- Real number with exponents
  : E number of all digits. number of fractional digits
  (9 to 24)
  (Number of all digits ≥ number of fractional digits+8)
- String
  : C number of digits
  (0 to 255)
- Binary number
  : B number of digits
  (1 to 8)
- Hexadecimal number
  : H number of digits
  (1 or 2)

(3) Examples

```
PRINT A$:C3,J:F1Ø.4
```

NOTES

- When number of digits is 0 for string, the character length becomes variable to output all actual length of the character string variable.
- A single space is included at the end of each PRINT statement provided that the FORMAT specifiers are capitalized. These spaces can be omitted by using a small-case FORMAT specifier instead of a capitalized FORMAT specifier (See Section 4, "Display(PRINT statement)" and "Reverse display (PRINTR statement)".)

Label

(1) Function

A jump address can be assigned indirectly by using a label with a line number in a statement such as GOTO or GOSUB.

(2) Format

```
Line number␣* label
Line number␣* label␣statement
```

- A label consists of up to eight alphanumeric characters starting with an uppercase alphabetic character. The label is prefixed with *.
- When multiple line numbers are defined with the same label, an error occurs during program execution.

(3) Examples

```
 10  INPUT A
 20  IF A=0 GOSUB * ABC1
 30  IF A<>0 GOSUB * ABC2
 40  GOTO 10
100  * ABC1
110  PRINT "OK!"
120  RETURN
200  * ABC2
210  PRINT "NG!"
220  RETURN
```

Basic Statements

Comment (REM statement)

(1) Function

This statement gives comments to program. These comments are not executed by the system and they have no effect on program execution.

Note: When a specific statement is described as a comment statement, it must be enclosed by a pair of double quotation marks(" ") as a character constant.

(2) Format

```
REM ["comment"] or
' [comment]
```

(3) Examples

```
1Ø  REM
2Ø  REM "Compute average"
3Ø  'Compute average
4Ø  A=1ØØ 'Initial set
```

Array declaration (DIM statement)

(1) Function

This statement declares arrays. Arrays must be one-dimensional or two-dimensional, and are restricted at a size as shown in paragraph (2) below according to the type of variable name.

(2) Format

```
DIM variable-name(array-size[,array-size])
    [,variable-name(array-size[,array-size])....]
```

Notes:

- The same variable name cannot be redefined as an array. A variable (that has been used as an independent variable) cannot be declared as an array.
- Error W225 will be generated when a two-dimensional array is referred to without the specification of two dimensions.
- Error W224 will be generated when a one-dimensional array is referred to as a two-dimensional array.
- The size limit of the declarable array is as follows. If the declared size exceeds these limits, ERROR 203 will be generated.

Character type ..... 1 to 255	Two dimensional array:		
Bit type ..... 1 to 8	One dimensional side	Two dimensional side	
Numeric type ..... 1 to 1024	1 to 1024	Character type	1 to 255
		Bit type	1 to 8
		Numeric type	1 to 1024

- For the numeric type, the program area will become insufficient; thus, it is impossible to define 1024 on both the one- and two-dimensional sides. In this case, ERROR 206 will be generated.
  The total number of array elements that can be declared (product of the number of one-dimensional array elements by the number of two-dimensional array elements) is not restricted because it depends on the capacity of empty memory.
- For the character array, ten characters long are automatically declared when no array is declared.
- For the bit type, array eight bits long are automatically declared when no array is declared.
- Error W224 occurs when individual elements are referred to (read or written) without the appropriate array declaration.

(3) Examples

```
DIM  CARR(1ØØ),A$(5,12)

DIM  I#(8),ALP$(4Ø)
```

(4) System variables which have been unconditionally declared as arrays.

```
XMA(*),XMB(*),
XMT(*),XMB(*),SMA(*),SMB(*),SMT(*),IMA(*),IMB(*),RMA(*),RMB(*)
```

NOTES

* is an array element of 0 to 500.

Initialization (CLEAR statement)

(1) Function

Initializes user-defined variables.

(2) Format

```
CLEAR
```

Note: When the CLEAR statement is executed, the array can be redefined since variables are re-initialized in a manner similar to that in which executing RESET is executed.

Substitution (LET statement)

(1) Function

This statement substitutes variables for constants, variables, and results of operations.

See Section 4, "Arithmetic operators".

(2) Format

```
[LET]variable=[([ constant  ])]
              [ variable    ]
              [ function    ]

[arithmetic operator[([ constant  ])] ...]
                    [ variable    ]
                    [ function    ]
                  + -
                  * /
                   !

[LET]character type variable=[ character string constant ]+
                             [ character type variable   ]
                             [ character string function ]

                             [ character string constant ]+ ...
                             [ character type variable   ]
                             [ character string function ]
```

Notes:

- Bits and characters cannot be used in operations.
- If a substitution statement is placed after an IF statement, LET cannot be omitted.

(3) Examples

```
LET A=B+C or A=B+C

IF X=Ø LET Y=1Ø
```

Branch (GOTO statement)

(1) Function

This statement changes the sequence of program execution to the statement of the specified line number.

(2) Format

```
GOTO line number or GOTO * label
```

Termination of execution (STOP statement)

(1) Function

This statement terminates program execution after displaying an execution termination message on the CRT screen as follows.

```
STOP IN  line  number
```

(2) Format

```
STOP
```

Note: Suspension specifications are ignored in STOP statements, since program execution is terminated.

Branch to subroutines (GOSUB statement)

(1) Function

This statement changes the program execution to the subroutine with the specified line number. When the RETURN statement is executed at the end of the subroutine, the program execution is returned to the statement following the GOSUB statement.

(2) Format

```
GOSUB line number or GOSUB * label
```

Note: Calling another subroutine during execution of a subroutine is referred to as "nesting". Up to 10 nesting levels are permitted.

Return from subroutines to main routine (RETMAIN statement)

(1) Function

When the RETMAIN command is used during program execution, control is returned to the highest level of the routine regardless of the nesting level.

(2) Format

```
RETMAIN
```

Note: If the RETMAIN command has been executed in the highest level of the routine, ERROR F213 occurs.

Return from subroutines (RETURN statement)

(1) Function

This statement returns program execution from the subroutine to the statement following the corresponding GOSUB statement.

(2) Format

```
RETURN
```

Decision (IF statement)

(1) Function

If the result of the relational operation is true, this statement executes the subordinate statement. For relational operators, see Section 4, "Relational operators".

(2) Format

```
IF { constant } relational operator { constant } statement
   { variable }                     { variable }
```

relational operator:

- =
- >< or <>
- >
- <= or =<
- <
- >= or =>

Notes:

- All statements including IF statements can be placed as subordinate statements.
- Relational operations can not be performed among numerical values, characters, and bits.
- If a substitution statement is placed after an IF statement, LET cannot be omitted.

(3) Examples

```
IF C=1 GOTO 1ØØ

IF ACH$=BCH$ PRINT ACH

IF C<1Ø IF C>=2Ø PRINT "ERROR"

IF C<1Ø LET C=1Ø
```

Repetitions start (FOR statement)

(1) Function

This program loop causes the program code (located between the FOR and NEXT) to be repeatedly executed, until the specified variable is equal to or greater than the specified end value.

Up to 10 nesting levels may occur within a FOR statement.

(2) Format

```
FOR numeric variable = { numeric constant } TO { numeric constant }
                       { numeric variable  }    { numeric variable  }
                              |                        |
                         Initial value            Ending value

[ STEP { numeric constant } ]
       { numeric variable  }
              |
   Increment (default value is 1)
```

Notes:

- Even if the initial value exceeds the end value, one operation cycle will be performed.
- NEXT statements may be used anywhere; however, for proper execution they must be properly positioned.

(3) Example

```
FOR C=1 TO 1ØØ
FOR T=TB TO TE STEP Ø.1
FOR D=-1 TO -1Ø STEP -1
NEXT D
NEXT T
NEXT C
```

Repeats

Repetition termination (NEXT statement)

(1) Function

This statement is used with its corresponding FOR statement to terminate the repeated operation.

(2) Format

```
NEXT numeric variable
```

Same variable as that specified in FOR statement

Key-input (INPUT statement)

(1) Function

This statement is used to assign data input from the front panel key to variables.
When the statement is executed, the following message is displayed on the CRT.

?

Input data after the display question mark ? via the numeric key of the front panel, then press [ENTER] key of the instrument.

Use commas (,) as delimiters of data if required.

(2) Format

```
INPUT ["displayed character string",] variable[,variable....]
```

Notes:

- If a real number is input for an integer variable, it is truncated under decimal point.
- If the input data length is smaller than that which has been declared, spaces are appended to the entry. If it is greater, the excess digits will be truncated.
- For numeric and bit type variables, spaces before and after the input value are ignored.
- Hexadecimal data cannot be input.
- Five variables can be specified .
- The ,(comma) and –(minus) are input by pressing the [kHz] key and the [MHz] key of the front panel, respectively.

(3) Examples

```
INPUT "COUNT=",C  → COUNT=? 123

INPUT C,A$,I#     → ? 123,Q,1Ø11Ø1
```

Display (PRINT statement)

(1) Function

This statement edits and displays data on the CRT screen.

Unformatted data is displayed with spaces added after its effective digits. The format name and output formats are shown in Table 4-2.

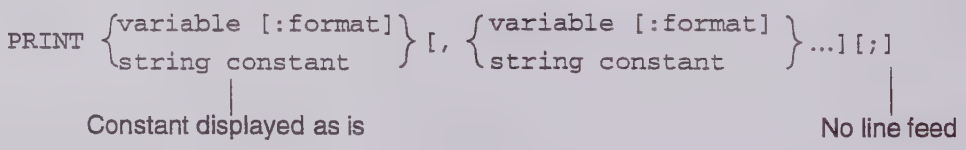
For the format, see Section 4, "Formats".

Line feed is disabled by adding " ; " at the end.

Table 4-2 Format Name and Output Format

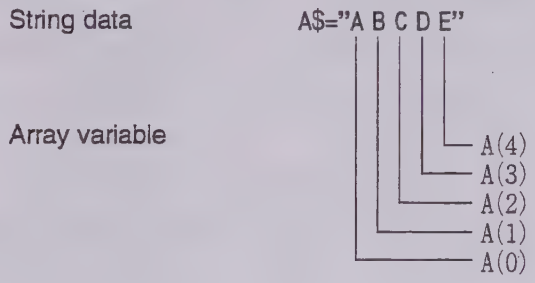
Format name	Output format
I	Zero-suppressed integer (Ex. ␣␣123)
F	Zero-suppressed integer and zero-suppressed fraction (Code digit exists.) (Ex. ␣␣123.45␣)
FP	Zero-suppressed integer and zero-suppressed decimal number (unsigned) (Ex. ␣123.45␣)
E	{ ␣ / - } Zero-suppressed fraction E [–] exponent (Ex. ␣1.23E–2␣)
C	String ... If the size of data is smaller than the specified format size, spaces are added; and if it is greater, the excess lower digits are truncated.
B / H	Zero-suppressed binary-number/hexadecimal-number string (Ex. ␣␣1011)

(2) Format

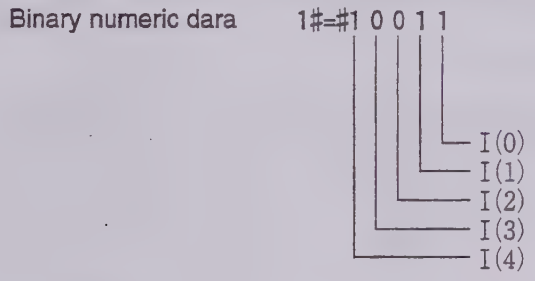


Notes:

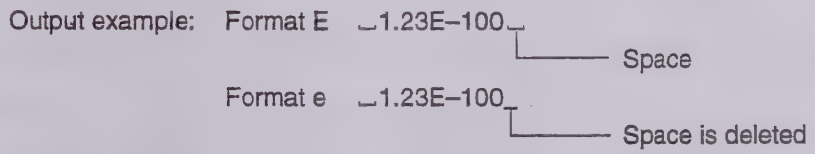
- Up to five variables or constants can be specified.
- Values which cannot be expressed are displayed as ***...*.
- A string-which is an array of character variables- is comprised as follows:



- A binary numeric variable- which is an array of binary digits- is comprised as follows:



- The last space can be deleted by using a lower-case format i, f, fp, e, c, b, or h instead of an upper-case format I, F, FP, E, C, B, or H.



- Only plus values are significant in format FP.

(3) Data and print output examples

Table 4-3 shows data and output examples.

Table 4-3 PRINT-Statement Output Example

Format	Data	Statement	Output
(None)	T=1234.45	PRINT␣T	123.45␣
	A$="ABCD"	DIM␣A$(5)	
		PRINT␣A$	ABCD␣␣
		PRINT␣A$(2)	C␣
	A$(Ø,)="AB"	DIM␣A$(3,2)	
	A$(1,)="CD"	PRINT␣A$(1,Ø)	C␣
	A$(2,)="EF"	PRINT␣A$(2,)	EF␣
I	T=1234.56	PRINT␣T:I6	␣␣1234␣
		PRINT␣T:I4	1234␣
		PRINT␣T:I3	***␣
F	T=-123.45	PRINT␣T:F6.1	-123.4␣
		PRINT␣T:F9.2	␣␣-123.45␣
		PRINT␣T:F9.3	␣-123.45Ø␣
	T=123456	PRINT␣T:F9.1	␣123456.Ø␣
		PRINT␣T:F5.1	*****␣
FP	T=123.45	PRINT␣T:FP6.1	␣123.4␣
		PRINT␣T:FP9.2	␣␣␣123.45␣
		PRINT␣T:FP9.3	␣␣123.45Ø␣
	T=123456	PRINT␣T:FP9.1	␣123456.Ø␣
		PRINT␣T:FP5.1	*****␣
E	T=-123.45	PRINT␣T:E1Ø.2	-1.23E2␣␣␣␣
		PRINT␣T:E13.5	-1.2345␣E2␣␣␣␣
		PRINT␣T:E15.7	-1.2345␣␣␣E2␣␣␣␣
	T=-Ø.12E1	PRINT␣T:E9.2	-1.2␣EØ␣␣␣␣
C	A$="F"	PRINT␣A$:C3	F␣␣␣
	A$="ABCDE"	DIM␣A$(5)	
		PRINT␣A$:C7	ABCDE␣␣␣
		PRINT␣A$:C3	ABC␣
		PRINT␣A$:C5	ABCDE␣
		PRINT␣A$(3):C3	D␣␣␣
	A$="ABCDEF"	DIM␣A$(6)	
		PRINT␣A$	ABCDEF␣
		PRINT␣A$(3)	D␣

Table 4-3 PRINT-Statement Output Example (Continued)

Format	Data	Statement	Output
B	I#=#1	PRINT␣I#:B1	1␣
		PRINT␣I#:B3	ØØ1␣
	I#=#1Ø11	DIM␣I#(4)	
		PRINT␣I#:B5	1Ø11␣␣
		PRINT␣I#:B3	Ø11␣
		PRINT␣I#(3):B3	1␣␣␣
		PRINT␣I#(Ø):B1	1␣
	I#=#1	PRINT␣I#	␣␣␣␣␣␣␣1␣
	I#=#1Ø11	DIM␣I#(4)	
		PRINT␣I#	1Ø11␣
	I#=#ØØØ1ØØ11	DIM␣I#(8)	
		PRINT␣I#	1ØØ11Ø1Ø␣
		PRINT␣I#(3)	1␣
	I#=#ØØØ1ØØ11	PRINT␣I#	␣␣␣1ØØ11␣
H	I#=#1	PRINT␣I#:H1	1␣
		PRINT␣I#:H2	␣1␣
	I#=#1Ø1Ø	DIM␣I#(4)	
		PRINT␣I#:H1	A␣
		PRINT␣I#:H2	A␣␣
	I#=#ØØØØ1Ø1Ø	DIM␣I#(8)	
		PRINT␣I#:H1	A␣
		PRINT␣I#:H2	␣A␣
	I#=#111Ø1Ø1Ø	DIM␣I#(8)	
		PRINT␣I#:H1	A␣
		PRINT␣I#:H2	EA␣
		PRINT␣I#(3):H1	1␣
		PRINT␣I#(3):H2	1␣␣
		PRINT␣I#(4):H1	Ø␣
		PRINT␣I#(4):H2	Ø␣␣
	I#=#ØØ11ØØ	DIM␣I#(6)	
		PRINT␣I#:H2	␣C␣
	I#=#11ØØ1Ø	PRINT␣I#:H2	32␣

Note

Example with the DIM statement means the array declaration is performed for the variable. If no DIM statement is marked, it means there is no array declaration for the variable.

Reverse display (PRINTR statement)

(1) Function

Edits data and displays the data on the screen in reverse mode.
See Section 4, "PRINT statement" for details.

(2) Format

```
PRINTR {variable [ : format]          } [,{variable [ : format]          }...][;]
       {character-string-constant     }   {character-string-constant     }
```

The constant is displayed as is.

No line feed

Notes:

- Only characters of character codes 0 to 127 can be displayed in reverse mode.
  PRINTR containing other character displays has the same function as that of PRINT. In this case, PRINTR displays characters in normal mode.
- A line in which characters of character codes 128 to 255 are displayed cannot be displayed in reverse mode.
  In this case, PRINTR has the same function as that of PRINT, and it displays characters in normal mode.

Positioning the cursor (LOCATE statement)

(1) Function

This statements specifies the cursor position on the screen. (Referred to at the upper left on the screen)

(2) Format

```
LOCATE(m,n)
    m   →   column position (1 to 40)
    n   →   line position (1 to 20)
```

Note: Both m and n are numeric constants or variables.

Data statement (DATA statement)

(1) Function

This statement defines numeric, bit and character constant to be read with the RDATA statement.

(2) Format

```
DATA, constant, constant, ••••••••
```

Note: Any number of parameters maybe input in a DATA statement provided that it does not exceed two lines. Further, different types of constants may be input in a single DATA statement.

Reading data (RDATA statement)

(1) Function

This statement reads values from the DATA statement and assigns them to variables.

(2) Format

```
RDATA variable, variable, ••••••••
```

Notes:

- Any number of parameters maybe assigned in an RDATA statement provided that it does not exceed 2 lines. Further, different types of constants may be input in a single RDATA statement.
- If the definition type in the DATA statement and the type of the substituted variable are incompatible at data reading with the RDATA statement, ERROR W208 will be generated.

Read specification of data statement (RESTORE statement)

(1) Function

This statement specifies the data statement to be read with the RDATA statement.

(2) Format

```
RESTORE [line number or * label]
```

Example :

```
10Ø  RESTORE 1ØØØ
11Ø  FOR I=Ø TO 1Ø
12Ø  RDATA A(I)
13Ø  NEXT I
      :
1ØØØ  DATA Ø,1,3,7,9,11,13,17,19,23,29
```

Note: When the RESTORE-statement parameter is omitted, the first data statement is used.

Setting measurement parameters (PUT and WRITE 1000 statements)

(1) Function

Sets the spectrum analyzer measurement parameters from the PTA.
The same messages as those set by remote control are used.
This command is also used when sending inquiry messages to the spectrum analyzer.

(2) Format

```
PUT character constant or character variable
WRITE 1ØØØ, variable or character constant [,variable or character constant]
```

1) PUT statement

- A message of the same format as remote control is described in operands.
- Only a character constant or character variable can be described in the operands.
- Only one constant or variable can be described.
- The format cannot be specified.
- When a fixed value is set at all times, the program can be simplified using this statement.

Examples :

```
PUT " CF 5ØØMHZ"
```

→ Set measurement parameter center frequency to 500 MHz.

```
PUT " CF?"
```

→ Send measurement parameter center frequency inquiry message.

2) WRITE 1000 statement

- A message of the same format as remote control is described in operands.
- Variables or character constants can be described in the operands.
- Up to five constants or variables can be described.
- When variables are used, the format can be specified.
- This statement is effective when setting is performed several times with only part of the control message being changed and when values treated as variables are set values in the program.

Examples :

```
F=5ØØ
WRITE 1ØØØ,"CF ",F,"MHZ"
```

→ Set measurement parameter center frequency to 500 MHz.

```
WRITE 1ØØØ,"CF?"
```

→ Send measurement parameter center frequency inquiry message.

Measurement parameter/data read (GET, COM and READ 1000 statements)

(1) Function

Reads the spectrum analyzer measurement parameters and the measured result from the PTA.
The same messages as those set by remote control are used.

(2) Format

```
GET "inquiry command?",input variable
COM "inquiry command?">input variable[, input variable]
READ 1000, input variable[, input variable] or
READ 1000, input variable[;]
```

1) GET statement

- An inquiry command can be sent and the response data can be read with one statement. Only one inquiry command can be described in one statement.
- Only a character constants or character variables can be described in the "inquiry command" parameters. Only one constant or variable can be specified. The format cannot be specified.
- The response data is stored in the input variable. When the response data contains a character, a character variable is specified. When the response data is numeric (numeric character) only, it may be a numeric variable or a character variable.
- When the response data consists of multiple data separated by a ",", everything up to the last data is stored in one variable as one data. Therefore, when a character variable is specified, if the array size is too small, all the response data may not be stored.
- Only one input variable can be specified. A ";" cannot be specified at the end of the statement.
- When the same inquiry command is always sent, the program can be simplified using this statement.

Example :

```
GET "CF?", A$
```

→ Send the center frequency inquiry message and store the response data in input variable A$.

2) COM statement

- An inquiry command can be sent and the response data can be read with one statement. However, only one inquiry command can be described in one statement.
- Character constant or character variable or character constant and character variable can be specified in the "inquiry command" parameter.
  The format can also be specified for variables.

- The response data is stored in the input variable. When the response data contains a character, a character variable is specified. When the response data is numeric (numeric character) only, it can be a numeric variable or character variable.
- Multiple variables can be described. When the response data consists of multiple data delimited by a ",", the delimited data are stored sequentially in the specified variables.
  However, array variables cannot be used as input variables.
- A ";" cannot be specified at the end of the statement.
- This statement is effective when reading is performed several times with only part of the inquiry message changed and when sending an inquiry message for a value treated as a variable in the program.

Example :

```
I=1
COM "MKML? ", I>ML
```

→ Send the 1st marker level inquiry message of the multimarker, and store the response data to input variable ML.

Note: The inquiry message for each level of the multimarker is specified by "MKML? n " (n: multimarker No.). This statement is useful for reading the level of each marker by changing only the value of n.

3) READ 1000 statement

- This statement reads the response data only. Therefore, it is effective only when a PUT or WRITE 1000 statement is used to send an inquiry message.
- The response data is stored in the input variable. When the response data contains a character, a character variable is specified. When the response data is numeric (numeric character) only, it can be a numeric variable or character variable.
- Multiple input variables can be described. When the response data consists of multiple data delimited by a ",", the delimited data is stored sequentially in the specified variables.
- When the response data is treated as one data, even when it consists of multiple data delimited by a ",", the entire response, including the ",", can be stored in one variable by specifying ";" at the end of the statement. In this case, only one input variable can be specified. Data delimited by a "," can also be read by specifying only one variable without a ";" at the end and executing this statement repeatedly.
- When there is no response data, "***" is output.

Example :

```
WRITE 1000,"CF? "
READ 1000, A$
```

→ Store the response data to the center frequency inquiry command in A$.

Program loading and execution (CHAIN statement)

(1) Function

This statement loads and executes a file in memory card.

(2) Format

```
CHAIN "file name"
```

Note: The RUN, CONT or STEP commands (set in the execution state) remain valid even after the CHAIN command is executed. Consequently, the lines at which execution is suspended also remain effective.

ENABLE EVENT statement

(1) Function

Enables the specified interrupt.

When the specified interrupt occurs, the program will branch to the event interrupt subroutine defined by the ON EVENT statement.

(2) Format

```
ENABLE EVENT I/O number,event 3,event 2,event 1,event 0
```

Notes:

- There are 2 types of I/O numbers: numeric variables and numeric constants.
- Events 0 to 3 can be numeric variables and constants, bit variables and constants, or hexadecimal constants.
- This statement can be executed directly.
- Events 0 to 3 indicate 32 bits of I/O interrupt events as shown below.
- The defined bits (b0 to b31) are enabled when "1" and disabled when "0".
- When the master bit (b31) was set to "1", all the defined conditions are valid regardless of the value of bits b0 to b30.

b31 b24	b23 b16	b15 b8	b7 b0
Event 3	Event 2	Event 1	Event 0

(3) Types of I/O interrupts

(a) Time-specification interrupts

Three kinds of time-specification interrupts are available.

1) DELAY

Generates an event interrupt after the specified time has elapsed.
The time can be specified as a remote control command or by a PUT or WRITE statement.

DELAY setting

"EDLY t" t: 0 to 3600 (s) 1 sec resolution

- Time counting starts from the time set by this command.
- When the time is reset during counting, counting restarts.
- If t=0 was set, counting is interrupted.
- There is no set value t inquiry command.

2) Time

Generates an event interrupt at the specified time.
The time can be specified as a remote control command or by a PUT or WRITE statement.

Time setting
"ETIM t_1, t_2, t_3"

t_1: Specifies the hour. (0 to 23)
t_2: Specifies the minute. (0 to 59)
t_3: Specifies the second. (0 to 59)

- When the time is reset during counting, counting restarts.
- There are no set value t_1, t_2, and t_3 inquiry commands.

3) Cycle

Generates an event interrupt at the specified cycle (time).
The cycle can be specified as a remote control command or by a PUT or WRITE statement.

Cycle setting

"ECYC t" t: 0 to 3600 (s) 0.1 sec resolution

- If t=0 was set, time counting is interrupted.
- There is no set value t inquiry command.

(b) Soft keys and data knob interrupt

1) Soft keys ([F1] to [F5])

When a PTA menu (3/4) [F1] to [F5] key (corresponding to system variables EX1 to EX5) is pressed, an event interrupt is generated. This also applies to the PTA keyboard [F1] to [F5] keys.

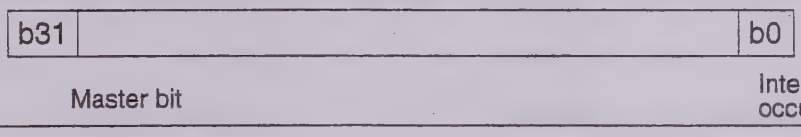
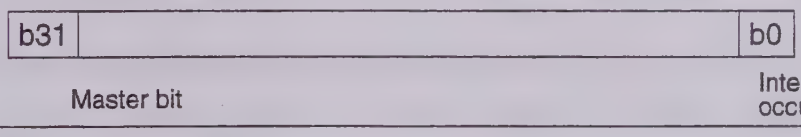
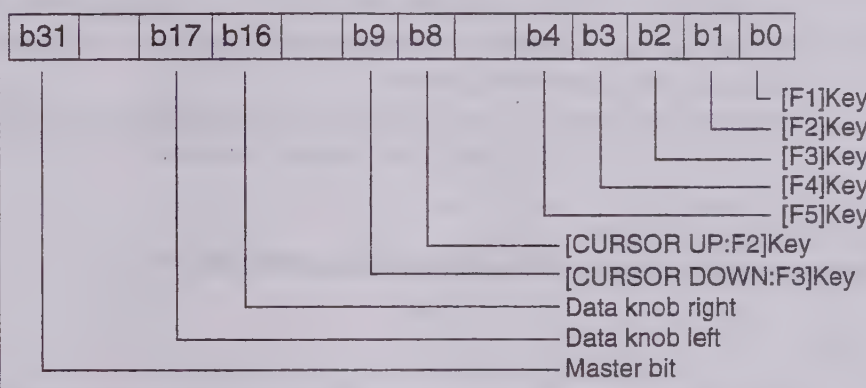
2) Cursor control keys

When the PTA menu (2/4) [CURSOR UP : F2] key or [CURSOR DOWN : F3] key is pressed, an event interrupt is generated.

3) Data knob

When the data knob is turned, an event interrupt is generated.
However, when the spectrum analyzer measurement parameter setting is effective, an event interrupt is not generated.

Clockwise and counterclockwise revolution can be detected.

I/O type	I/O number	Contents
Clock (DELAY)	1	
Clock (TIME)	2	
Clock (CYCLE)	3	
SOFT KEY, data knob	11	 b31 b17 b16 b9 b8 b4 b3 b2 b1 b0 [F1]Key [F2]Key [F3]Key [F4]Key [F5]Key [CURSOR UP:F2]Key [CURSOR DOWN:F3]Key Data knob right Data knob left Master bit

DISABLE EVENT statement

(1) Function

Disables the specified interrupt.

(2) Format

```
ENABLE EVENT I/O number[,event 3,event 2,event 1,event 0]
```

Notes:

- There are 2 types of I/O number: numeric variables and numeric constants.
- Events 0 to 3 can be numeric variables and constants, bit variables and constants, or hexadecimal constants.
- Events 0 to 3 may be omitted. When omitted, all interrupt events will be disabled.
- This statement can be directly executed.
- The defined bits are disabled when "1" and retain their previous enable/disable state when "0". However, master bit (b31) setting is meaningless. (Don't care)

ON EVENT statement

(1) Function

Registers the subroutine to branch to when the specified interrupt event occurs.

(2) Format

```
ON EVENT I/O number, line number(or * label)
```

Notes:

- There are 2 types of I/O number: numeric variables and numeric constants.
- This statement can be executed directly.
- The function STATUS (M) is used as the interrupt event identifier. For more details, see Section 4, "Functions", (5) Dedicated functions.

RETINT statement

(1) Function

Returns from the event interrupt subroutine.

(2) Format

```
RETINT
```

Notes:

- If any other return command is executed to return from an event interrupt subroutine, an execution termination error (F243) will be generated.
- If the RETINT command is executed for other than event interrupt, an execution termination error (F251) will be generated.
- It is possible to branch to a normal subroutine (GOSUB ... RETURN) from the event interrupt subroutine.

Character size specification (DCHSIZE statement)

(1) Function

Specifies the display character size at system subroutine DCH execution.

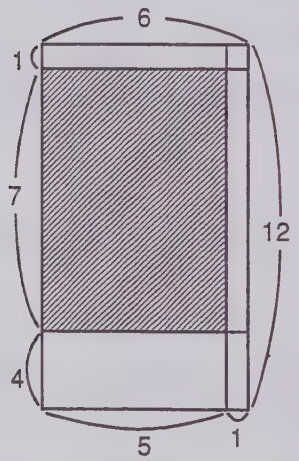
(2) Format

```
DCHSIZE Character size number
```

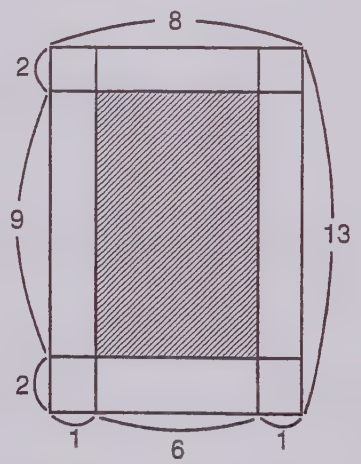
Character size number	
0	Small font
1	Medium font

- The patterns of small/medium character fonts are shown below:

Small font



Medium font



The units are dots on the CRT.

- The display character size can not be changed by PRINT statement, etc.
- Initialized by the RESET command.

Home position (HOME statement)

(1) Function

This statement moves the cursor to the home position (upper left).

(2) Format

```
HOME
```

Delete (ERASE statement)

(1) Function

This statement deletes statements after the line with the cursor.

(2) Format

```
ERASE
```

Note: When only the PTA screen is erased from the display, the screen is only partially erased. To erase the screen entirely, use the system subroutine CFL (see Section 5, "CFL subroutine").

Time wait (WAIT statement)

(1) Function

This statement is used to wait for a specified time period.

(2) Format

```
WAIT { Numeric variable }
     { Numeric constant }
```

Waiting time (unit: second, Ø.Ø1 s resolution)

System subroutine execution (CALL statement)

(1) Function

This statement is used to execute system subroutines.
For details of system subroutines, see Section 5, "System Subroutines".

(2) Format

```
CALL system subroutine name[(parameter[,parameter...])]
```

ON ERROR statement

(1) Function

Registers the subroutine to branch (interrupt) to when an error occurs.

(2) Format

```
ON ERROR line number(or * label)
```

Notes:

- Execution is halted when an error occurs during the execution of an error processing subroutine.
- If there is an error statement right after the line where the error occurred, only the error statement will be executed.
- If the error is an execution termination error, no interrupt will occur.
- If an error occurs during data input with the INPUT statement, no interrupt will occur.
- The function ERRREAD (m) identifies the error code and line the error occurred. For details, see Section 4, "Dedicated functions".
- Multiple interrupts with event interrupts are possible.
- The error occurred during an error interrupt processing is not applied.

OFF ERROR statement

(1) Function

Removes the registered subroutine to branch (interrupt) when an error occurs. No error interrupt will occur while after executing this command.

(2) Format

```
OFF ERROR
```

RETERR statement

(1) Function

Returns from an error interrupt.

Continues from the statement following the statement where the error occurred.

(2) Format

```
RETERR
```

Notes:

- If the RETURN or RETMAIN commands are used to return from an error interrupt ,an execution termination error (F243) will result.
- If the RETINT command is executed to return from an error interrupt, an execution termination error (F251) will result.
- If the RETERR command is executed when there is no error interrupt, an execution termination error (F252) will result.
- It is possible to branch to a normal subroutine (GOSUB … RETURN) from the event interrupt subroutine.

RETRY statement

(1) Function

Returns from an error interrupt.
Execution is retried from the statement on which error occurred.

(2) Format

```
RETRY
```

Notes:

- If the RETURN or RETMAIN commands are used to return from an error interrupt, an execution termination error (F243) will result.
- If the RETINT command is executed to return from an error interrupt, an execution termination error (F251) will result.
- If the RETRY command is executed when there is no error interrupt, an execution termination error (F252) will result.
- It is possible to branch to a normal subroutine (GOSUB ... RETURN) from the event interrupt subroutine.

RESUME statement

(1) Function

Returns from an error interrupt.
Continues from the specified line.

(2) Format

```
RESUME line number(or *label)
```

Notes:

- If the RETURN or RETMAIN commands are used to return from an error interrupt, an execution termination error (F243) will result.
- If the RETINT command is executed to return from an error interrupt, an execution termination error (F251) will result.
- If a command other than the RESUME command is executed when there is no error interrupt, an execution termination error (F252) will result.
- It is possible to branch to a normal subroutine (GOSUB ... RETURN) from the event interrupt subroutine.

GIVEUP statement

(1) Function

Returns from an error interrupt.
Halts program execution.

(2) Format

```
GIVEUP
```

Notes:

- If the RETURN or RETMAIN commands are used to return from an error interrupt, an execution termination error (F243) will result.
- If the RETINT command is executed to return from an error interrupt, an execution termination error (F251) will result.
- If the GIVEUP is executed when there is no error interrupt, an execution termination error (F252) will result.
- It is possible to branch to a normal subroutine (GOSUB … RETURN) from the event interrupt subroutine.

Error branch (ERROR statement)

(1) Function

To continue execution after warning-error generation, an ERROR statement can be used. Multiple lines can be used for ERROR statements.
See Section 8, "ERROR Statement" for details.

(2) Format

```
ERROR(error number,program line or *label to be executed next)
```

Error main (ERRMAIN statement)

(1) Function

This statement branches to the highest level routine when an error that allows execution to continue (error code beginning with the letter W) is generated while the program was running.

(2) Format

```
ERRMAIN(error number)
```

Notes:

- When an ERRMAIN statement was executed in the highest level routine, the error code becomes F213.
- See Section 8, "ERRMAIN Statement" for details.

Data input 1 (READ statement)

(1) Function

This statement is used to receive data from a device connected to the RS-232C or GPIB through the specified port.

(2) Format

```
READ address,input variable[,input variable....]
READ address,variable[;]
```

- When ";" is not added at the end of the statement, commas (",") in the received data are assumed to be data delimiters and are stored in each variable.
- When ";" is added at the end of the statement, commas (",") are not assumed to be data delimiters and everything up to the data terminator is stored in one variable.

Data input 2 (BREAD statement)

(1) Function

This statement is used to receive one byte of binary data from a device connected to the RS-232C or GPIB through the specified port. When the specified port is a device port, this statement cannot be executed.

(2) Format

```
BREAD address,input variable[,input variable....]
```

Data input 3 (WREAD statement)

(1) Function

This statement is used to receive one word of binary data from a device connected to the RS-232C or GPIB through the specified port. The data is stored in the input variable as high byte to low byte in sending order. When the specified port is a device port, this statement cannot be executed.

(2) Format

```
WREAD address,input variable[,input variable....]
```

Data output 1 (WRITE statement)

(1) Function

This statement sends data to a device connected to the RS-232C/GPIB/parallel (centronics) through the specified port.

(2) Format

```
WRITE address,variable[:format][,variable[:format]...][;]
```

- The output data can also be a character constant.
- When ";" is added at the end of the statement, a terminator is not output.
- The output destination depends on the addressing method and GPIB port mode (system controller/device).

Data output 2 (BWRITE statement)

(1) Function

This statement sends one byte of binary data to a device connected to the RS-232C/GPIB/parallel (centronics) through the specified port. When the specified port is a device port, this statement cannot be executed.

(2) Format

```
BWRITE address,variable[,variable...]
```

Notes:

- Neither format nor ";" can be specified.
- The terminator is not output.

Data output 3 (WWRITE statement)

(1) Function

This statement sends one word (two bytes) of binary data in order of high byte to low byte to a device connected to the RS-232C/GPIB/parallel (centronics) through the specified port. When the specified port is a device port, this statement is not executed.

(2) Format

```
WWRITE address,variable[,variable...]
```

Notes:

- Neither format nor ";" can be specified.
- The terminator is not output.
- When a one- or two-digit value is used (e.g. 5 or 17) for an address, the value becomes the address of the device connected to the port specified by the PORT command as a remore control command (Indirect Port specification). However, when a three-digit value (e.g. 105 or 217) is used, the first digit becomes the port address and the lower two digits become the address of the device connected to the port (Direct Port specification).
- The lower two digits of the address at indirect or direct port specification have no meaning in the RS-232C and parallel (centronics). However, these digits should still be specified for form' s sake.

  Example:

`WRITE␣5,"ABC" .....`	Data is sent to address 5 through the port specified by the PORT command (indirect port specification).
`READ␣1ØØ,A$ .....`	Data is input from a device connected to port No. 1 (RS-232C) (direct port specification).
`WRITE␣2Ø5,"ABC" .....`	Data is sent to address 5 through port No. 2 (GPIB) (direct port specification).
`WRITE␣3ØØ,"ABC" .....`	Data is sent to a device connected to port No.3 (parallel (centronics)) (direct port specification).

These address specifications are effective for the WRITE, BWRITE, WWRITE, READ, BREAD, WREAD and LISTG statements.

The relationship between the port specification command and controller port is as follows:

	Indirect port specification	Direct port specification		
	WRITE 5	WRITE 1Ø5	WRITE 2Ø5	WRITE 3Ø5
At power-ON or after "PORT␣1" execution	*1 The RS-232C port is a controller port.	*1 The RS-232C port is a controller port.	The GPIB port is a controller port.	*1 The paralell (centronics) port is the controller port.
After "PORT␣2" execution	The GPIB port is a controller port.	*1 The RS-232C port is a controller port.	The GPIB port is a controller port.	*1 The parallel (centronics) port is the controller port.
After "PORT␣3" execution	*1 The parallel (centronics) port is the controller port.	*1 The RS-232C port is the controller port.	The GPIB port is the controller port.	*1 The parallel (centronics) port is the controller port.

*1: Addresses specified in the RS-232C, parallel (centronics) have no meaning. However, these addresses should still be specified for form' s sake.

Data writing to the dual port memory (WDPM statement)

(1) Function

This statement writes data to the dual port memory.
See Section 7, "Dual Port Memory" for details.

(2) Format

```
WDPM memory number,variable[:format][,variable[:format]....]
```

Notes:

- The output data can also be character constants.
- ";" cannot be specified.
- This statement can be executed regardless of the GPIB mode (system controller/device).

Data reading from the dual port memory (RDPM statement)

(1) Function

This statement reads data from the dual port memory.
See Section 7, "Dual Port Memory" for details.

(2) Format

```
RDPM memory number,input variable[,input variable ....]
```

- ";" cannot be specified.
- When data delimited by "," is input, multiple input variables are specified.

S.O.S (SOS)

(1) Function

This statement is displayed in the statement where a syntax error is generated during program loading.

(2) Format

```
SOS
```

Notes:

- A statement with SOS added is treated as a comment statement, the same as a REM statement, but when the program is run, it is treated as a syntax error.
- Line-number errors are treated as syntax errors (W6) and SOS is not displayed.

Setting the pseudorandom number sequence (RNDMIZE statement)

(1) Function

Sets a new initial value of a pseudorandom number sequence generated by the RND function.

(2) Format

```
RNDMIZE
```

Note: If this statement is not executed, the RND function in the program generates the same pseudo-random number sequence each time the program is executed.

Calling the PTA library (CALLIB statement)

(1) Function

This statement calls the specified PTA library.

(2) Format

```
CALLIB "PTA library name" [,parameter]
```

Numeric variable or constant (up to 10 parameters)

Alphanumeric string with up to 8 characters starting with a capital alphabet
Characters available for the 2nd character on :
- Under bar
- Capital alphabet : A to Z
- Small alphabet : a to z
- Numeral : 0 to 9

However, small alphabets are converted to capitals.

- The specified PTA library is called out. When the STOP statement is executed in the called PTA library, the system returns to the program where the CALLIB statement was executed.
- Up to 10 parameters can be sent to the called PTA library. In this case, parameter values are assigned to the local variables specified by the PARASET statement of the called PTA library. (See PARASET.)
- Nesting of the PTA library by the CALLIB statement is available up to 10 times.

Note: The PTA library, from the start line to the STOP statement, is counted as one program unit. (The STOP statement may come in the middle of the program.) The CALLIB statement calls this program unit.

Removing the PTA library from program memory (REMOVE statement)

(1) Function

This statement removes the specified PTA library from the program memory.

(2) Format

```
REMOVE ["PTA library name"]
```

Alphanumeric string with up to 8 characters starting with a capital alphabet
Characters available for the 2nd character on :
Under bar
Capital alphabet : A to Z
Small alphabet : a to z
Numeral : 0 to 9
However, small alphabets are converted to capitals.

- The specified PTA library is removed from the program memory. However, it is not possible to specify the PTA library in execution (or an error is generated if specified).
- When this function is directly executed without specifying a program, all the PTA libraries in the memory are removed.
- When the PTA library specified as the object of the program execution and edition commands is removed by the EDITLIB command, the specification of the EDITLIB command is cleared.

Clearing common variables (COMCLEAR statement)

(1) Function

This statement clears all the common variables residing in the memory.

(2) Format

```
COMCLEAR
```

- All the common variables residing in the memory are cleared.
- When this statement is executed in the nested PTA library, an error is generated.

Setting CALLIB parameter values (PARASET statement)

(1) Function

This statement sets the parameter values sent from the CALLIB statement to the specified local variables.

(2) Format

```
PARASET Parameter[,parameter]
                 |
        Up to 10 real-number local variables
```

- Parameters sent from the side that called the PTA library are set to local variables. Only the real-number local variable can be used. When common and other variables are specified, an error is generated at input. When the call side of the PTA library does not send parameters, the variable value is set to be zero.

Loading the PTA library file LOADLIB statement)

(1) Function

This statement loads the function-specified PTA library file.

(2) Format

```
LOADLIB "File name"
```

"File name" : Alphanumeric string with up to 6 characters starting with a capital alphabet
Characters available for the 2nd character on :
Capital alphabet : A to Z
Small alphabet : a to z
Numeral : 0 to 9
However, small alphabets are converted to capitals.

- The PTA library file saved in the memory card is loaded. If a PTA library named the same as one already existing in the memory is loaded, the content of the existing PTA library is replaced with that of the newly loaded PTA library.
- It is not possible to load the file in which a PTA library named the same as one in execution is saved.

SECTION 5

EXTENDED PTL

TABLE OF CONTENTS

System Variables 5-3

System Subroutines 5-5

- CER and CRN subroutines 5-7
- CFL subroutine 5-8
- DCH subroutine 5-9
- DLN subroutine 5-11
- DRC subroutine 5-13
- DCR subroutine 5-15
- DAR subroutine 5-17
- DEF subroutine 5-19
- OPNI, OPNO and FDEL subroutines 5-20
- DALD and DASV subroutines 5-21
- CLS subroutine 5-22
- IFC subroutine 5-22
- OPNITF, OPNOTF, FDEFTF subroutines 5-23
- DALDTF, DASATF subroutines 5-24
- CLSTF subroutine 5-25
- RSV subroutine 5-26
- TCT subroutine 5-27
- DEV subroutine 5-27
- GST subroutine (GST) 5-28
- Interface control subroutine (GPIB and RS-232C) 5-29
- PNLU and PNLL subroutine 5-31
- COPY subroutine 5-32
- CONV subroutine 5-33
- SWLG subroutine 5-34

System Functions 5-35
MAX function 5-38
MIN function 5-39
BNDL, BNDH, MESL, and MESH functions 5-40
RPL1 and RPL2 functions 5-42
RPL3 function 5-43
PEKL and PEKH functions 5-44
POLL and POLH functions 5-46
PLRH, PLLH, PLRL and PLLL functions 5-48
PFRQ function 5-50
SUM function 5-51
PSML and PSMH functions 5-52
DPOS and DNEG functions 5-54

SECTION 5
EXTENDED PTL

There are system variables, system functions, and system subroutines in the extended PTL.

The extended PTL can execute operations and evaluation of measurement results, and control external devices.

System Variables

PTA provides system variables with pre-defined names in addition to user-defined variables. Using these system variables, the measured data can be read.

Variable name	Number of array elements	Purpose	Data meaning	Read/ Write
EX1	---	Corresponding to F1 key	Numbers 0 and 1 are switched alternately each time the F1 key is pressed.	R/W
EX2	---	Corresponding to F2 key	Numbers 0 and 1 are switched alternately each time the F2 key is pressed.	R/W
EX3	---	Corresponding to F3 key	Numbers 0 and 1 are switched alternately each time the F3 key is pressed.	R/W
EX4	---	Corresponding to F4 key	Numbers 0 and 1 are switched alternately each time the F4 key is pressed.	R/W
EX5	---	Corresponding to F5 key	Numbers 0 and 1 are switched alternately each time the F5 key is pressed.	R/W
EX6	---	Corresponding to etc key of each hierarchy	0 to 3: Switches a PTA function key hierarchy (*)	R/W

* Soft-key menus can be changed by inputting 0, 1, 2 and 3 to the system variable EX6, as shown below.

However, EX6 is disabled when the PTA menus are not being executed.

Variable name	Number of array elements	Purpose	Data meaning	Read/ Write
DTØ	---	Time setting/reading (year: Gregorian calendar)	1960 to 2059	R/W
DT1	---	Time setting/reading (month)	0 to 12	R/W
DT2	---	Time setting/reading (date)	0 to 31	R/W
DT3	---	Time setting/reading (hour)	0 to 23	R/W
DT4	---	Time setting/reading (minute)	0 to 59	R/W
XMA	501	Waveform memory of TRACE-A	Waveform data in 0.01dBm unit	R/W
XMB	501	Waveform memory of TRACE-B	Waveform data in 0.01dBm unit	R/W
XMG	501	Waveform memory of TRACE-BG	Waveform data in 0.01dBm unit	R/W
XMT	501	Waveform memory of TRACE-Time	Waveform data in 0.01dBm unit	R/W
SMA	501	Submemory A	–32768 to 32767: 2-byte integer/1 point	R/W
SMB	501	Submemory B	–32768 to 32767: 2-byte integer/1 point	R/W
SMT	501	Submemory Time	–32768 to 32767: 2-byte integer/1 point	R/W
IMA	501	Image memory A	–32768 to 32767: 2-byte integer/1 point	R/W
IMB	501	Image memory B	–32768 to 32767: 2-byte integer/1 point	R/W
RMA	501	Real number memory A	8-byte floating point real number/1 point	R/W
RMB	501	Real number memory B	8-byte floating point real number/1 point	R/W

	EX6 = 0	EX6 = 1	EX6 = 2	EX6 = 3
F1	RUN	PLIST	F1 *	YES
F2	STOP	CURSOR UP	F2 *	NO
F3	CONT	CURSOR DOWN	F3 *	(None)
F4	RESET	LOAD	F4 *	(None)
F5	PTA OFF	RUN	F5 *	(None)
F6	etc (1/4)	etc (2/4)	etc (3/4)	etc (4/4)

* Display characters can be defined with DEF subroutine.

System Subroutines

The MS2665C/67C/68C PTA has dedicated subroutines, called the system subroutines, executed by the CALL statement.

The system subroutines are shown below :

■ Display subroutines

- Displayed item erase : `CALL CER(M)`
- Screen restore : `CALL CRN(M)`
- Screen erase : `CALL CFL(M)`
- Character-string display : `CALL DCH(X,Y,text,M[,N])`
- Straight-line display : `CALL DLN(XØ,YØ,X1,Y1,M[,N])`
- Square display : `CALL DRC(XØ,YØ,X1,Y1,M[,N])`
- Circle display : `CALL DCR(X,Y,R,M[,N])`
- Arc-line display : `CALL DAR(XØ,YØ,RØ,W1,W2,M1[,M3])`
- Soft-key label registration: `CALL DEF(M,text)`

■ File-operation subroutines

- File open (read) : `CALL OPNI␣character string variable (or character constant)`
- File open (write) : `CALL OPNO␣character string variable (or character constant)`
- File delete : `CALL FDEL␣character string variable (or character constant)`
- Data load : `CALL DALD variable`
- Data save : `CALL DASV variable`
- File close : `CALL CLS`

■ GPIB subroutine (GPIB port only)

- Interface clear : (Changeover to system controller port) `CALL IFC`
- Service request : `CALL RSV(M)`
- Take controller : `CALL TCT(M)`
- Changeover to device port : `CALL DEV`

■ Interface subroutine

- Status byte reading : `CALL GST(port number,address,input variable)`
- Interface control : `CALL GPIB(port number,control item number)`

■ Panel subroutines

- Front-panel operation lock : `CALL PNLL(Ø)`
- Front-panel operation lock cancellation : `CALL PNLU(Ø)`

■ Waveform memory subroutine

- Memory copy : `CALL COPY(MØ,M1)`
- Data conversion : `CALL CONV(K,MØ,M1,PØ,P1[,D])`
- Frequency axis logarithm conversion : `CALL SWLG(K,MØ,M1)`

NOTES

If parameters specified in each subroutine are outside the specified range, an error occurs and no graphic data is plotted.

CER and CRN subroutines

(1) Function

The CER/CRN subroutines perform erasure and display restoration of the character string, graph, scale, marker, etc. on the CRT screen.

(2) Format

```
CALL␣CER(MØ) ····· Erases items MØ
CALL␣CRN(MØ) ····· Restores items MØ display
```

M0	Item
0	Marker frequency, level, AT, RB
1	RLV, ST, VB
2	Frequency
3	Menu, data input area
4	Sweep marker
5	Scale line, Y-axis scale
6	Waveform
7	Markers, zone
8	Message in scale
9	Title, trace item, trigger switch, sweep status
10	All items above

Notes:

- See Section 1, "Screen Configuration of PTA" for the screen details.
- A numeric constant or numeric variable is used for M0.
- When clear/display return was performed with this subroutine, the state is held until it is reset by this subroutine or until the PTA is turned off.

CFL subroutine

(1) Function

This subroutine erases display items of each frame constituting the screen.

(2) Format

```
CALL_CFL(M1)
```

M1 (Frame No.)	Display item
0	Waveform background
1	PTA screen
2	Scale line
3	Waveform display 2
4	Waveform display 3
5	Parameter
6	Display line
7	Trigger indicator
8	Marker zone
9	Template/mask standard line
10	Multi-marker No.
11	(Not used)
12	Marker/marker value display
13	Menu background
14	Menu characters
15	Setting and parameter characters, error message

Notes:

- A numeric constant or numeric variable is used for M1.
- This subroutine temporarily clears the screen.
  Therefore, when the display condition is reestablished; for example, when measurement parameter values are changed, or when characters and patterns are displayed; they are displayed.
- See Section 1, "Screen Configuration of PTA" for the screen details.

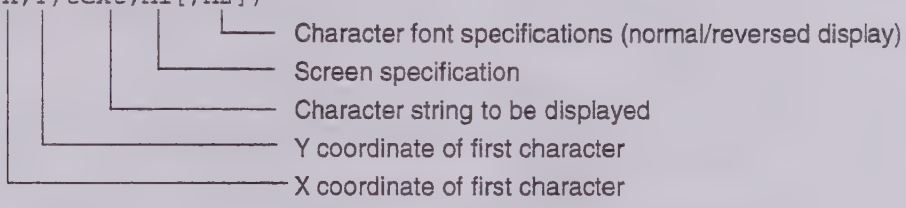
DCH subroutine

(1) Function

Displays a character string. (Referred to at the bottom left on the screen)

(2) Format

```
CALL␣DCH(X,Y,text,M1[,M2])
```


Character font specifications (normal/reversed display)
Screen specification
Character string to be displayed
Y coordinate of first character
X coordinate of first character

M1 (Frame No.)	Display item
0	Waveform background
1	PTA screen
2	Scale line
3	Waveform display 2
4	Waveform display 3
5	Parameter
6	Display line
7	Trigger indicator
8	Marker zone
9	Template/mask standard line
10	Multi-marker No.
11	(Not used)
12	Marker/marker value display
13	Menu background
14	Menu characters
15	Setting and parameter characters, error message

M2	Display mode
0	Normal display
1	Reverse diaplay

■ Range of each parameter

Font	First X coordinate (X)	First Y coordinate (Y)	Maximum No. ogf characters of string (text)
Small font	0 to 314	0 to 228	54
Medium font	0 to 312	0 to 227	40

Notes:

- The first X coordinate and Y coordinate specify the lower-left corner of the character.
- Numeric constants or numeric variables are used for X, Y, M1, and M2. "text" is a character constant or character variable.
- M2 is omissible and it is assumed to be 0 if omitted.
- The character size (small font/medium font) can be set with the DCHSIZE statement.
  DCHSIZE 0: Small font
  DCHSIZE 1: Medium font
- See Section 1, "Screen Configuration of PTA" for the screen details.

DLN subroutine

(1) Function

This subroutine displays a straight line (sectional line).

(2) Format

```
CALL_DLN (XØ,YØ,X1,Y1,M1[,M3])
```

- M3: Line type
- M1: Screen type
- Y1: Y coordinate at point P1 (Ø to 239)
- X1: X coordinate at point P1 (Ø to 319)
- YØ: Y coordinate at point PØ (Ø to 239)
- XØ: X coordinate at point PØ (Ø to 319)

M1 (Frame No.)	Display item
0	Waveform background
1	PTA screen
2	Scale line
3	Waveform display 2
4	Waveform display 3
5	Parameter
6	Display line
7	Trigger indicator
8	Marker zone
9	Template/mask standard line
10	Multi-marker No.
11	(Not used)
12	Marker/marker value display
13	Menu background
14	Menu characters
15	Setting and parameter characters, error message

M3	Line type
0	Displays solid line
1	Erases solid line
2	Displays dashed line
3	Erases dashed line

Notes:

- A numeric constant or numeric variable is used for X0, Y0, X1, Y1, M1, and M3.
- M3 is omissible and it is assumed to be 0 if omitted.
- See Section 1, "Screen Configuration of PTA" for coordinate details.

DRC subroutine

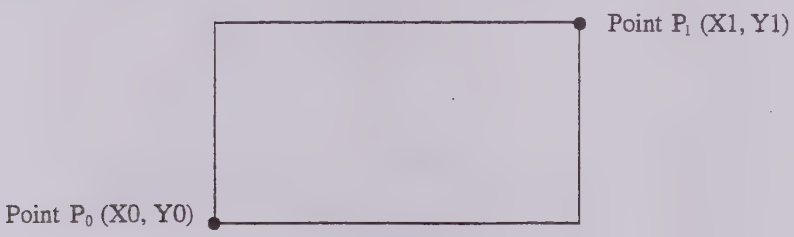
(1) Function

This subroutine displays a square based on a diagonal line between two specified points.

(2) Format

```
CALL␣DRC(XØ,YØ,X1,Y1,M1[,M3])
```

- M3: Line type
- M1: Screen type
- Y1: Y coordinate at point P1 (Ø to 239)
- X1: X coordinate at point P1 (Ø to 319)
- YØ: Y coordinate at point PØ (Ø to 239)
- XØ: X coordinate at point PØ (Ø to 319)



M1 (Frame No.)	Display item
0	Waveform background
1	PTA screen
2	Scale line
3	Waveform display 2
4	Waveform display 3
5	Parameter
6	Display line
7	Trigger indicator
8	Marker zone
9	Template/mask standard line
10	Multi-marker No.
11	(Not used)
12	Marker/marker value display
13	Menu background
14	Menu characters
15	Setting and parameter characters, error message

M3	Line type
0	Displays solid line
1	Erases solid line
2	Displays dashed line
3	Erases dashed line

Notes:

- A numeric constant or numeric variable is used for X0, Y0, X1, Y1, M1, and M3.
- M3 is omissible and it is assumed to be 0 if omitted.
- See Section 1, "Screen Configuration of PTA" for coordinate details.
- No display is performed if P0 (X0, Y0) and P1 (X1, Y1) are at the same axis.

DCR subroutine

(1) Function

This subroutine displays a circle.

(2) Format

```
CALL␣DCR(X,Y,R,M1[,M3])
```

- Line type
- Screen type
- Radius (1 to 675)
- Y coordinate of center (–239 to 478)
- X coordinate of center (–319 to 638)

M1 (Frame No.)	Display item
0	Waveform background
1	PTA screen
2	Scale line
3	Waveform display 2
4	Waveform display 3
5	Parameter
6	Display line
7	Trigger indicator
8	Marker zone
9	Template/mask standard line
10	Multi-marker No.
11	(Not used)
12	Marker/marker value display
13	Menu background
14	Menu characters
15	Setting and parameter characters, error message

M3	Line type
0	Displays solid line
1	Erases solid line
2	Diaplsys dashed line
3	Erases dashed line

Notes:

- Numeric constants or numeric variables are used for X, Y, R, M1, and M3.
- M3 is omissible and it is assumed to be 0 if omitted.
- See Section 1, "Screen Configuration of PTA" for coordinate details.

DAR subroutine

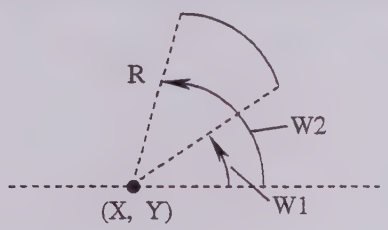
(1) Function

Displays an arc.

(2) Format

```
CALL␣DAR(X,Y,R,W1,W2,M1[,M3])
```

- Line type
- Screen type
- Plot end angle (–18Ø.ØØ to 18Ø.ØØ deg.)
- Plot start angle (–18Ø.ØØ to 18Ø.ØØ deg.)
- Radius (1 to 675)
- Coordinate Y of center (–239 to 478)
- Coordinate X of center (–319 to 638)



M1 (Frame No.)	Display item
0	Waveform background
1	PTA screen
2	Scale line
3	Waveform display 2
4	Waveform display 3
5	Parameter
6	Display line
7	Trigger indicator
8	Marker zone
9	Template/mask standard line
10	Multi-marker No.
11	(Not used)
12	Marker/marker value display
13	Menu background
14	Menu characters
15	Setting and parameter characters, error message

M3	Line type
0	Displays solid line
1	Erases solid line
2	Displays dashed line
3	Erases dashed line

Notes:

- Numeric constants or numeric variables are used for the X, Y, R, W1, W2, M1, and M3.
- M3 is omissible and it is assumed to be 0 if omitted.
- See Section 1, "Screen Configuration of PTA" for coordinate details.

DEF subroutine

(1) Function

Registers a menu label (name) in the soft key menu.

When the PTA menu (3/4) is displayed, the labels registered by this subroutine are displayed.

(2) Format

```
CALL␣DEF(M,text)
          │  └── Name of 3Ø characters maximum
          └───── Soft-key number (1 to 6)
```

Notes:

- M is a numeric constant or numeric variable.
- "text" is a character constant or character variable.
- The labels registered by this subroutine remain valid until the PTA is turned off.

OPNI, OPNO and FDEL subroutines

(1) Function

Opens a data file to write data to and read data from a memory card and deletes an existing data file.

(2) Format

```
CALL␣OPNI␣character string-variable(or character constant)
Open data read
CALL␣OPNO␣character string-variable(or character constant)
Open data write
CALL␣FDEL␣character string-variable(or character constant)
Delete data file
```

Notes:

- The data file name always begins with a % symbol and is followed by 6 or less alphanumeric characters including %.
- Do not remove the memory card while opening the data file in it.
- This subroutine cannot be used with the PTA program/library files on the memory card.

DALD and DASV subroutines

(1) Function

The DALD subroutine reads data saved in the memory card, and the DASV subroutine saves data to the memory card.

(2) Format

```
CALL␣DALD␣input variable:Read data from data file
CALL␣DASV␣variable       :Write data to data file
```

Notes:

- Data files are created as sequential files.
  Therefore, read them in the order in which they were written.
- Different types of data (for example, numeric type and character type) can be stored in one data file. However, when the type when the data was written and the type of input variable when the data was read cannot be assigned, an error is generated.

(3) Program example

```
 1Ø REM "*** DATA FILE ***"
 2Ø CALL OPNO"%DATA"
 3Ø FOR C=2 TO 1Ø
 4Ø D=C*C
 5Ø CALL DASV D
 6Ø NEXT C
 7Ø CALL CLS
 8Ø CALL OPNI"%DATA"
 9Ø FOR C=2 TO 1Ø
1ØØ CALL DALD D
11Ø PRINT "RES.=",D
12Ø NEXT C
13Ø CALL CLS
14Ø STOP
```

Writing into file

Reading from file

Excuted result

```
RES.=4
RES.=9
RES.=16
RES.=25
RES.=36
RES.=49
RES.=64
RES.=81
RES.=1ØØ
```

CLS subroutine

(1) Function

This subroutine closes the open data file.
Used for both write and read.

(2) Format

```
CALL␣CLS
```

IFC subroutine

(1) Function

When this subroutine is executed, the GPIB port becomes the system controller and outputs an "interface clear" signal to devices connected to the GPIB bus.

(2) Format

```
CALL␣IFC
```

Note: When CALL␣IFC is executed from the PTA, GPIB becomes the "connection port for peripheral devices" of the conditions for interface port connection. Accordingly, if GPIB has been set as the connection port for the external controller and the printer/plotter, the "connection port for the external controller" and the "connection port for the printer/plotter" becomes "no connection (NONE)".

OPNITF, OPNOTF, FDELTF subroutines

(1) Function

Opens a text file to write and to read text data from a memory card and deletes an existing text data file. This file can be read and written as plain text file on personal computer. File attribute (.txt) is added automatically.

(2) Format

CALL␣OPNITF	character string-variable (or character constant) Open text data read
CALL␣OPNOTF	character string-variable (or character constant) Open text data write
CALL␣OPNOTF	character string-variable (or character constant) Delete data file

Notes:

- The text data file name is followed by 6 or less alphanumeric characters.
- Do not remove the memory card while opening the next data file in it.
- This subroutine cannot be used with the PTA program/library files on the memory card.

DALDTF, DASATF subroutines

(1) Function

The DALD subroutine reads text data saved in the memory card, and the DASV subroutine saves data to the memory card.

(2) Format

CALL␣DALDTF	string-variable :	Read data from text data file
CALL␣DASVTF	string-variable (character constant) :	Write data to next data file

Notes:

- When DALDTF subroutine is executed, 1 line is read from text data file, and that is stored string-variable. If the text data is longer than variable length, the text data is cut by variable length.
  If string-variable is not used, an error is generated.
- When DASVTF subroutine is executed, 1 line is text data is stored to data file. If string-variable is not used, an error is generated.

(3) Program example

```
1Ø   CALL OPNOTF "RWTEST"
2Ø   FOR I=Ø TO 25
3Ø   D$=CHR$ (64+I)
4Ø   CALL DASVTF D$
5Ø   NEXT I
6Ø   CALL CLSTF
7Ø   FOR I=Ø TO 25
8Ø   CALL DALDTF D$
9Ø   PRINT D$
1ØØ  NEXT I
11Ø  CALL CLSTF
12Ø  STOP
```

CLSTF subroutine

(1) Function

This subroutine closes the open data file.
Used for both write and read.

(2) Format

```
CALL␣CLSTF
```

RSV subroutine

(1) Function

This subroutine sends the service request to the controller when the GPIB port (the first interface) is used as a device port.

(2) Format

```
CALL␣RSV(M)
```

M	PTA Event Status Register							
	MSB							LSB
0	×	×	×	×	0	0	0	1
1	×	×	×	×	0	0	1	0
2	×	×	×	×	0	0	1	1
3	×	×	×	×	0	1	0	0
4	×	×	×	×	0	1	0	1
5	×	×	×	×	0	1	1	0
6	×	×	×	×	0	1	1	1
7	×	×	×	×	1	0	0	0
8	×	×	×	×	1	0	0	1
9	×	×	×	×	1	0	1	0

(× means don't-care bit which does not change.)

The PTA event status register is defined as the extended status of Status-Byte bit 1.

Therefore, setting the left-described data (into the PTA Even Status Register) indirectly sets Status-Byte bit 1 as a summary bit.

The RQS bit (bit 6) is set as the logical AND of each Status-Byte bits to issue a service request to the controller.

The GPIB commands (used to read the Status Byte and PTA Event Status Register from the external controller) are *STB? and ESR1 ?, respectively.

Notes:

- A numeric constant or numeric variable is used for M.
- This subroutine is effective only when the GPIB port is connected with the external controller (the device port mode).

TCT subroutine

(1) Function

This subroutine causes controlling right to be passed to another device provided that the GPIB port is used as a system controller port.

(2) Format

```
CALL_TCT(M)
         |
         Address of device to which control right is passed.
```

Notes:

- M is the GPIB address from 0 to 30, and a numeric constant or numeric variable is used.
- This subroutine is effective only when the GPIB port is a system controller port.

DEV subroutine

(1) Function

This subroutine causes the GPIB port to become a device port when it has previously been used as the system controller.

(2) Format

```
CALL_DEV
```

Note: When the CALL DEV subroutine is executed from PTA, the "connection port for the external controller" of the conditions for interface port connection becomes GPIB. Accordingly, if GPIB has been set as the connection port for peripheral devices and the printer/plotter, the "connection port for peripheral devices" and the "connection port for the printer/plotter" becomes "no connection (NONE)".

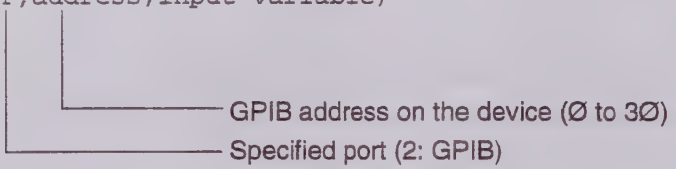
GST subroutine (GST)

(1) Function

When the GPIB port is set as the connection port for the external controller, a serial poll is executed to the device specified by address, and the status value is read and stored as an input variable.

(2) Format

```
CALL␣GST(P,address,input variable)
```



Notes:

- The read status value will be stored in the input variable. Input variable can be either a real-number, integer, or bit type variable.
- This subroutine is effective only when the GPIB port is a system controller port.
- This subroutine cannot be executed on the RS-232C/parallel (centronics).

Interface control subroutine (GPIB and RS-232C)

(1) Function

The "Interface Clear" (IFC), "Remote" (REN), "Local" (LCL), "Device Clear" (DCL), "Local Rockout" (LLO), and "Device Trigger" (DTR) are sent, and "Return to Local" (RTL) is set from the specified port.

(2) Format

`CALL␣GPIB(P,Ø)`	Sends IFC
`CALL␣GPIB(P,1[,address])`	Sends REN
`CALL␣GPIB(P,2)`	Sends RTL
`CALL␣GPIB(P,3[,address])`	Sends LCL
`CALL␣GPIB(P,4[,address])`	Sends DCL
`CALL␣GPIB(P,5)`	Sends LLO
`CALL␣GPIB(P,6,address)`	Sends DTR

P : Specified port No. (RS-232C: 1, GPIB: 2, Parallel (centronics): 3)

Address: GPIB device address of Ø to 3Ø

Notes:

- P and address are numeric constants or numeric variables.
- The actions of each subroutine are described below.

IFC :
- The IFC line is turned on for 100 É sec. The interface functions of all connected devices are initialized.
- Initialization is executed only for the corresponding interface functions. This code does not affect device functions.
- All talkers and listeners are not released.
- This does not affect the SRQ line.
- If the system passes control of the GPIB port to other controllers with the CALL TCT (m) command, control will be automatically returned to the PTA when execution is finished.
- This subroutine terminates normally without performing any processing for the RS-232C.

REN:
- When [, address] is omitted, the REN line is turned ON. Afterwards when the device is set to listener, it will assume remote control status.
- When [, address] is specified, the REN line is turned on. The device specified by [, address] will be identified as the listener and assume remote control status.
- Can be executed only when the specified port is a system controller port.
- This subroutine terminates normally without performing any processing for the RS-232C.

Notes: (Continued)

RTL:
- When the GPIB port is identified as the device, the PTA assumes the local control status. (This has the same effect as pressing the [LOCAL] key.)
- Only "2" can be specified as the port No.

LCL:
- When [, address] is omitted, the REN line is turned off. All devices assume local control status.
- When [, address] is specified, all listeners are released. After that, the device specified by [, address] is selected as the listener and assumes local control status. The REN line does not change.
- Can be executed only when the specified port is a system controller port.

DCL:
- When [, address] is omitted, "DCL" is sent and all device functions on the GPIB are initialized.
- When [, address] is specified, (Selected Device Clear) is sent and the device function specified by [, address] is initialized.
- Can be executed only when the specified port is a system controller port.

LLO:
- Disables the remote to local switching function of all devices on the GPIB. You will not be able to switch the device to local with the [Local] key on the panel.
- Switching is possible with the REN and LCL commands from the PTA.
- This mode can be exited with the LCL command in which the [, address] is omitted.
- Can be executed only when the specified port is a system controller port.

DTR:
- Triggers the specified device. The specified device begins the predetermined operation.
- Can be executed only when the specified port is a system controller port.
- This subroutine terminates normally without performing any processing for the RS-232C.

PNLU and PNLL subroutine

(1) Function

Sets LOCK/UNLOCK of the front panel when PTA is on.

(2) Format

```
CALL␣PNLU(Ø)        unlocks front panel.
CALL␣PNLL(Ø)        Locks front panel.
```

Note: The front-panel soft keys [F1] to [F6], [Shift], [Local] ,and numeric keys cannot be lock-out.

COPY subroutine

(1) Function

This subroutine copies the data in a specified waveform memory (copy source) to another waveform memory (copy destination). For example, use of the sub memory permits measurement in parallel with data processing.

(2) Format

```
CALL␣COPY(MØ,M1)
          │ └─ Destination memory
          └─── Source memory
```

M0, M1	Memory	System variable name	Type
0	Measurement memory	XMA ( )	Integer (0.01 dBm unit)
1	Measurement memory	XMB ( )	Integer (0.01 dBm unit)
2	Submemory a	SMA ( )	Integer (0.01 dBm unit)
3	Submemory b	SMB ( )	Integer (0.01 dBm unit)
4	Image memory a	IMA ( )	Integer
5	Image memory b	IMB ( )	Integer
6	Real number memory a	RMA ( )	Real number
7	Real number memory b	RMB ( )	Real number
8	Measurement memory	XMT ( )	Integer
9	Measurement memory	XMB ( )	Integer
10	Sub memory	SMT ( )	Integer

Notes:

- M0 contents are copied in M1. M0 contents are not changed. Previous contents of M1 are lost.
- A numeric constant or numeric variable is used for M0 and M1.
- Data cannot be copied between integer memory and real number memory.

CONV subroutine

(1) Function

This subroutine converts the measurement data of the measurement memory and performs the operation between memories.

(2) Format

```
CALL CONV(K,MØ,M1,PØ,P1[,D])
```

- Sub-data
- Area specification (PØ<P1)
- Destination memory
- Source memory
- Function specification

K			Conversion (operation) function
0	Integer (0.01 dBm)	→	Real number (dBm)
1	Real number (dBm)	→	Integer (0.01 dBm)
2	Integer (0.01 dBm)	→	Real number (mW) M1 (x)=10 ↑ (M0 (x)/1000)
3	Real number (mW)	→	Integer (0.01 dBm) M1 (x)=INT (1000*LOG_{10} (M0 (x)))
4	ADD M1=M0+D		
5	SUB M1=M0–D		
6	MUL M1=M0*D		
7	DIV M1=M0/D		
8	ADDA M1=M1+M0+D		
9	SUBA M1=M1–M0+D		
10	Running average (running average every D points, M1 (n)= $\frac{1}{D}\sum_{k=n-\frac{D-1}{2}}^{n+\frac{D-1}{2}}$ M0 (k)) (D is odd number)		

Notes:

- When K is assumed to be 0 to 3, use the memory number 0 to 5, 8 or 9 for the memory called "integer", and use the memory number 6 or 7 for the memory called "real number".
- P0 and P1 are numeric constants or numeric variables from 0 to 500.
- D is a numeric constant or numeric variable. Its default is D=0.
- When K is 10, $(P0-\frac{D-1}{2}) \geq 0$ and $(P1+\frac{D-1}{2}) \leq 500$ must be satisfied.

SWLG subroutine

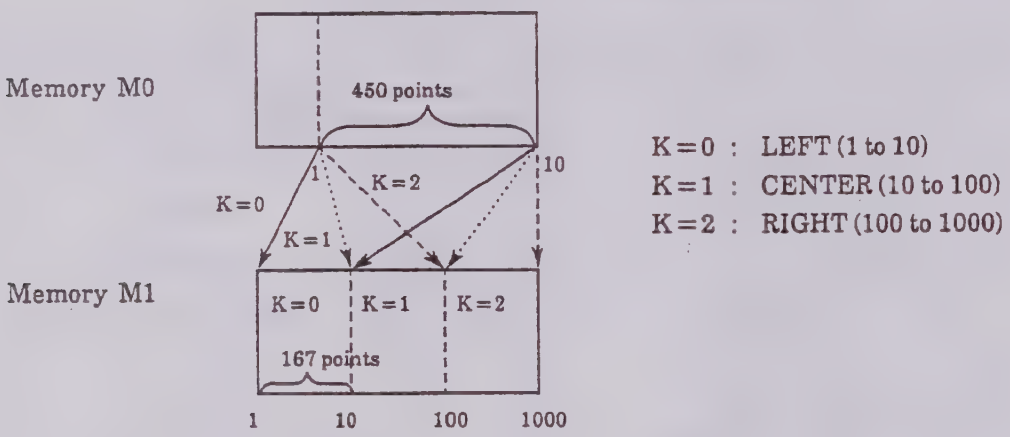
(1) Function

This subroutine arranges the data of the specified memory so that the frequency axis is LOG display and then transfers it.

(2) Format

```
CALL_SWLG(K,MØ,M1)
```

- M1 — Destination memory
- MØ — Source memory
- K — Function specification



K=0 : LEFT (1 to 10)
K=1 : CENTER (10 to 100)
K=2 : RIGHT (100 to 1000)

The memory M0 data is a measured value obtained by an ordinary (linear) sweep.
The frequency axis LOG for 3 decades can be displayed in memory M1 by sweeping three times by changing the frequency and by executing the SWLG subroutine three times.

Note: The M0 and M1 must be combined within the integer memories, or real M0 and M1 must be combined in the real number memories.

System Functions

The system functions can extract and calculate special points in the waveform data, with the waveform memory as the objective. Therefore, there is a function result value.

System function		Function
Maximum value	MAX(M, P0, P1 N)	Returns the maximum value between P0 to P1
Minimum value	MIN(M, P0, P1, N)	Returns the minimum value between P0 to P1
Frequency at specified measured value (1)	BNDL(M, P0, L, N)	Starts search from P0 and returns the frequency of the supecified measured value
Frequency at specified measured value (2)	BNDH(M, P0, L, N)	Starts search from P0 and returns the frequency at the supecified measured value
Frequency at specified measured value (3)	MESL(M, P0, L, N)	Starts search from P0 and returns the frequency of the supecified measured value
Frequency at specified measured value (4)	MESH(M, P0, L, N)	Starts search from P0 and returns the frequency of the supecified measured value
Ripple 1	RPL1(P0, P1, N [, R])	Obtains ripple 1 between P0 to P1
Ripple 2	RPL2(P0, P1, N [, R])	Obtains ripple 2 between P0 to P1
Ripple 3	RPL3(P0, P1, N [, R])	Obtains ripple 3 between P0 to P1
Peak 1	PEKL(M, P0, L, N [, R])	Starts search from P0 and returns peak value
Peak 2	PEKH(M, P0, L, N [, R])	Starts search from P0 and returns peak value
Pole 1	POLL(M, P0, L, N [, R])	Starts search from P0 and returns pole (dip) value
Pole 2	POLH(M, P0, L, N [, R])	Starts search from P0 and returns pole (dip) value
Inflection top value 1	PLRH(M, P0, N [, R])	Starts search from P0 and returns adjacent inflection maximum
Inflection top value 2	PLLH(M, P0, N [, R])	Starts search from P0 and returns adjacent inflection maximum
Inflection bottom value 1	PLRL(M, P0, N [, R])	Starts search from P0 and returns adjacent inflection minimum
Inflection bottom value 2	PLLL(M, P0, N [, R])	Starts search from P0 and returns adjacent inflection minimum

(Continued)

System function		Function
Frequency specified point	PFRQ(P0)	Returns frequency of P0 point
Total	SUM(P0, P1, N)	Returns total of the memory contents between P0 to P1
Addition search 1	PSML(M, P0, L, N)	Successively adds from P0 and returns a point with the specified value
Addition search 2	PSMH(M, P0, L, N)	Successively adds from P0 and returns a point with the specified value
Decision 1	DPOS(M, P0, P1, N1, N2)	Compares and decides the size of the memory contents
Decision 2	DNEG(M, P0, P1, N1, N2)	Compares and decides the size of the memory contents

Notes:

- Since the waveform memory is the objective of the system functions, the input values (P0 and P1) to each function are specified as points on all the waveform memories.
- P0, P1, L, N and R are input parameters indicated by a numeric constant or numeric variable.
- M is an output parameter indicated by a variable.
- N, N1 and N2 are parameter which specify the waveform memory. It is a numeric constant or numeric variable.

N, N1, N2	Memory	System variable name	Type
0	Measurement memory TRACE-A	XMA ( )	Integer
1	Measurement memory TRACE-B	XMB ( )	Integer
2	Submemory a	SMA ( )	Integer
3	Submemory b	SMB ( )	Integer
4	Image memory a	IMA ( )	Integer
5	Image memory b	IMB ( )	Integer
6	Real number memory a	RMA ( )	Real number
7	Real number memory b	RMB ( )	Real number
8	Measurement memory TRACE-TIME	XMT ( )	Integer
9	Measurement memory TRACE-BG	XMG ( )	Integer
10	Sub-memory t	SMG ( )	Integer

- [,R] can be omitted. When omitted, R is assumed to be 0.
- P0 and P1 specify the points in the waveform memory. Their setting range is 0 to 1001.
- P0 and P1 used in the system functions always specify the points in the measurement memories.

MAX function

(1) Function

This function obtains the maximum value in the specified measurement memory area and the measurement point at the maximum value.

(2) Format

```
MAX(M,PØ,P1,N)
```

- N — Measurement memory specification
- P1 — Area Specification (PØ<P1)
- PØ — Measurement point at maximum value
- M — Maximum value

Note: If there is more than one point with the same maximum value, the first point of the maximum value is stored in M.

(3) Program example: Obtains maximum level in measurement memory TRACE-A.

```
1Ø  REM "MAX(M,PØ,P1,N)"
2Ø  GMAX=MAX(M,Ø,5ØØ,Ø)
3Ø  GMAX=GMAX*Ø.Ø1
4Ø  PRINT "Maximum Level=",GMAX,"dBm"
5Ø  STOP
Maximum Level=-2Ø.45dBm
```

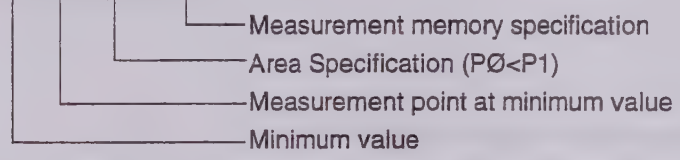
MIN function

(1) Function

This function obtain the minimum value in the specified measurement memory area and the measurement point at the minimum value.

(2) Format

```
MIN(M,PØ,P1,N)
```



Note: If there is more than one point with the same minimum value, the first minimum value point is stored in M.

(3) Program example: Obtains minimum level in measurement memory TRACE-B.

```
1Ø  GMIN=MIN(M,Ø,5ØØ,1)
2Ø  GMIN=GMIN*Ø.Ø1
3Ø  PRINT "Min Level=",GMIN,"dBm at",M
4Ø  STOP
```

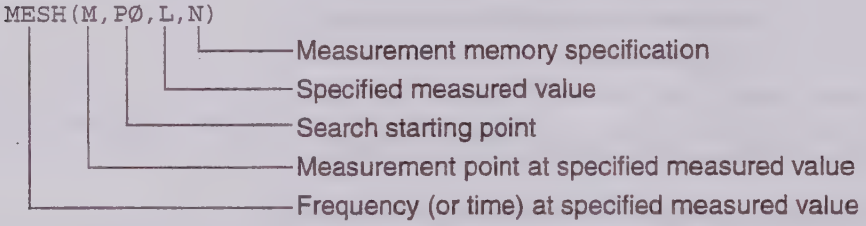
BNDL, BNDH, MESL, and MESH functions

(1) Function

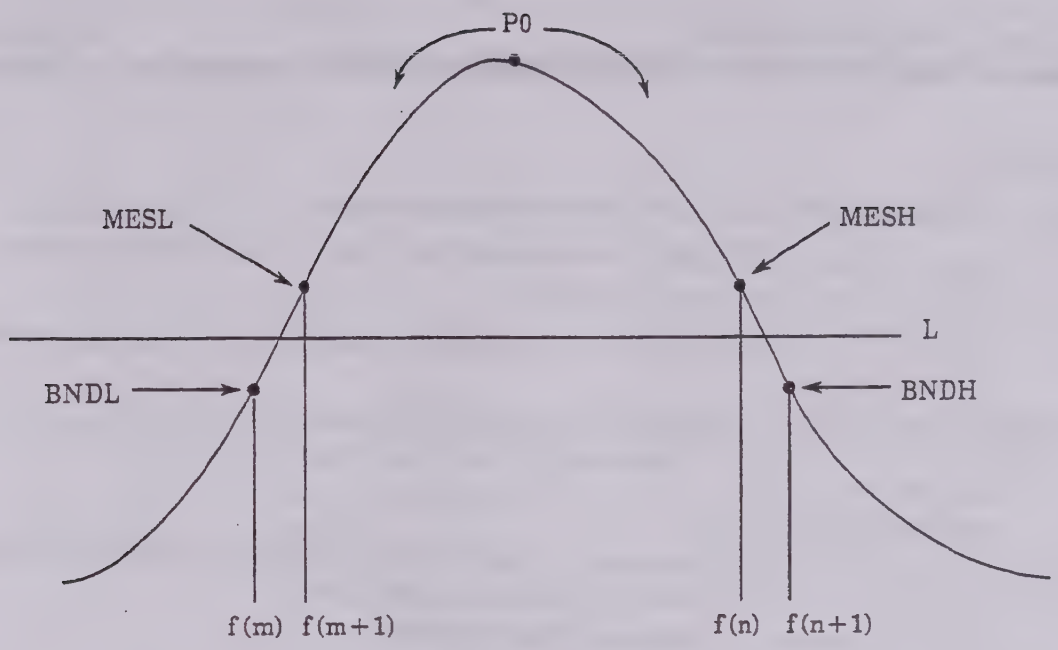
These functions obtain the frequency at the specified measured value by searching from a starting point in the specified memory.

(2) Format

```
BNDL(M,PØ,L,N)
BNDH(M,PØ,L,N)
MESL(M,PØ,L,N)
MESH(M,PØ,L,N)
```


Measurement memory specification
Specified measured value
Search starting point
Measurement point at specified measured value
Frequency (or time) at specified measured value

- When N is specified to 0, 2, 4, 6, 7
  Find the frequency of the specified measurement value from the TRACE-A setting frequency.
- When N is specified to 1, 3, 5, 7
  Find the frequency of the specified measurement value from the TRACE-B setting frequency.
- When N is specified to 8, 10
  Find the time of the specified measurement value from the TRACE-TIME setting time.
- When N is specified to 9
  Find the frequency of the specified measurement value from the TRACE-BG setting time.



Note: If there is no specified measured value in BNDL and MESL, M is assumed to be 0; in BNDH and MESH, M is assumed to be 1001.

(3) Program example: Obtains bandwidth at level of –20 dBm in A channel memory, searching from center.

```
1Ø  L=-2ØØØ ..... indicates –20 dBm
2Ø  FL=BNDL(ML,25Ø,L,Ø)
3Ø  FH=BNDH(MH,25Ø,L,Ø)
4Ø  BW=(FH-FL)/1ØØØ
5Ø  PRINT "BW=",BW,"KHz"
6Ø  STOP
```

RPL1 and RPL2 functions

(1) Function

These functions obtain ripple 1, and 2 in the specified memory area.

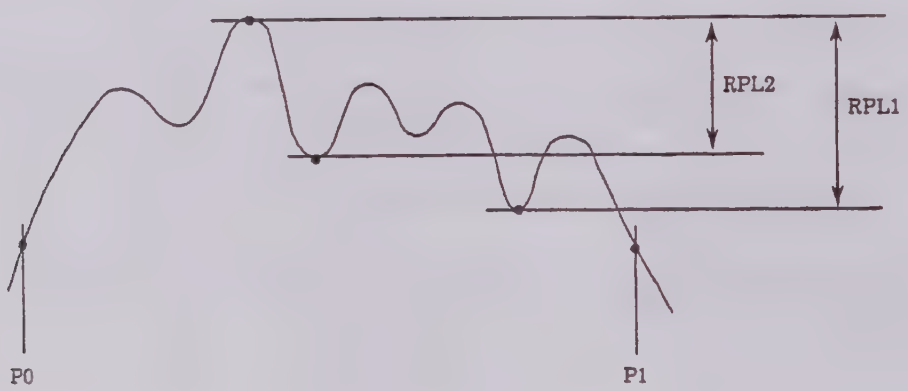
Ripple 1: This is the difference between the maximum value of the inflection top value and the minimum value of the inflection bottom value.

Ripple 2: This is the maximum difference between the adjacent inflection top and bottom values.

(2) Format

```
RPL1(PØ,P1,N[,R])
RPL2(PØ,P1,N[,R])
```

- R — Resolution of difference between adjacent inflection top and bottom values
- N — Measurement memory specification
- PØ, P1 — Area specifications (PØ<P1)
- RPL1 / RPL2 — Ripple



Notes:

- If the difference between the adjacent inflection top and bottom values is smaller than R, the ripple is not obtained.
- N which specifies the measured memory must be from 0 to 5, 8 or 9. (No real number memory can be used.)

(3) Program example: Obtains Ripple 1 between the measurement points 100 and 300 in measurement memory TRACE-A, where resolution is 0.2 dB.

```
1Ø  RP=RPL1(1ØØ,3ØØ,Ø,2Ø,) ..... R=20 when resolution is 0.2 dB
2Ø  RP=RP/1ØØ
3Ø  PRINT "RPL1=",RP,"dB"
4Ø  STOP
```

RPL3 function

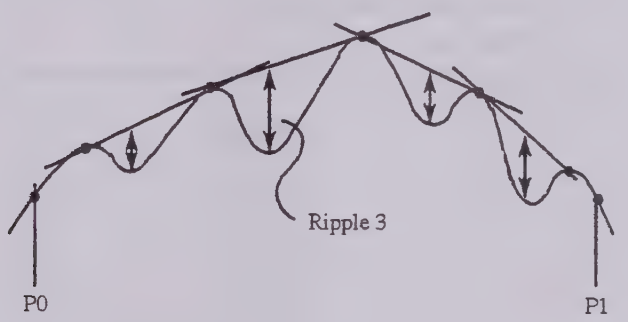
(1) Function

This function obtains the maximum difference between the adjacent tangent at the inflection top and inflection bottom value (ripple 3) in the specified memory area as shown a figure below.

(2) Format

```
RPL3(PØ,P1,N[,R])
```

- R — Resolution of difference between adjacent inflection top and bottom values
- N — Measurement memory specification
- PØ,P1 — Area psecification (PØ<P1)
- RPL3 — Ripple



Notes:

- If the difference between the adjacent inflection top and bottom values is smaller than R, the ripple is not obtained.
- N which specifies the measured memory must be from 0 to 5, 8 or 9. (No real number memory can be used.)

(3) Program example:

Obtains Ripple 3 between the measurement points 50 and 450 in the measurement memory TRACE-B, where resolution is 0.1 dB.

```
1Ø  RP=RPL3(5Ø,45Ø,1,1Ø,)
2Ø  RP=RP/1ØØ
3Ø  PRINT "RPL3=",RP,"dB"
4Ø  STOP
```

PEKL and PEKH functions

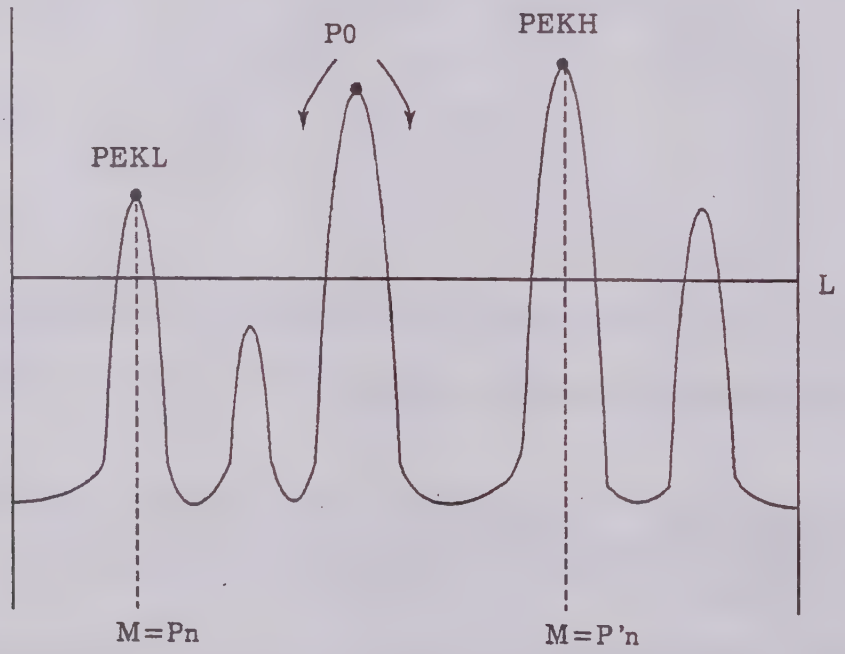
(1) Function

These functions find the first peak and its measured point, which is larger than the specified measured value in the measurement area, by searching from a starting point in the specified memory.

(2) Format

```
PEKL(M,PØ,L,N[,R])
PEKH(M,PØ,L,N[,R])
```

- R: Resolution for difference between adjacent inflection top and bottom value
- N: Memory specification
- L: Specified measured value
- PØ: Search starting point
- M: Peak measurement point
- PEKL/PEKH: Peak level



Notes:

- If the peak cannot be found with the PEKL function, M is assumed to be 0, and the measured value at point 0 is PEKL.
- If the peak cannot be found with the PEKH function, M is assumed to be 500, and the measured value at point 1001 is PEKH.
- N which specifies the measured memory must be from 0 to 5, 8 or 9. (The real number memory cannot be used.)
- If the difference between adjacent inflection top and bottom values is smaller than R, the inflection top is not the peak.

(3) Program example: Obtains peak level higher than –50 dBm searched left of the measurement point 200 in measurement memory TRACE-A, where resolution is 2 dB.

```
1Ø  PLEV=PEKL(M,2ØØ,-5ØØØ,Ø,2ØØ)
2Ø  PLEV=PLEV/1ØØ
3Ø  PRINT "Peak Level=",PLEV,"dBm at",M
4Ø  STOP
```

POLL and POLH functions

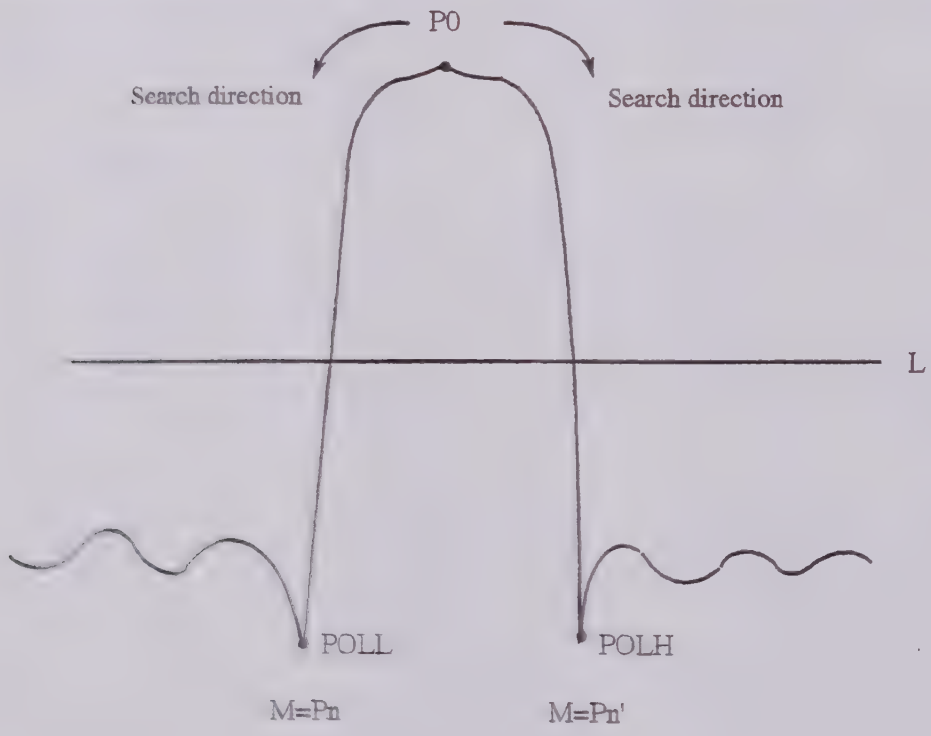
(1) Function

These functions obtain the pole and its measurement point, which is smaller than the specified measured value in the measurement area, by searching from a starting point in the specified memory.

(2) Format

```
POLL(M,PØ,L,N[,R])
POLH(M,PØ,L,N[,R])
```

- Resolution for difference between adjacent inflection top and bottom values
- Measurement memory specification
- Specified measured value
- Search starting point
- Measurement point at pole
- Pole level



Notes:

- If pole cannot be obtained in POLL function, M is assumed to be 0, and the measured value at point 0 is POLL.
- If pole cannot be obtained in POLH function, M is assumed to be 1001, and the measured value at point 500 is POLH.
- N which specifies the measured memory must be from 0 to 7, 8 or 9. (No real number memory can be used.)
- If the difference between adjacent inflection top and bottom values is smaller than R, the inflection top is not the pole.

(3) Program example: Obtains pole level lower than –60 dBm searched left of the measurement point 250 in measurement memory TRACE-A, where resolution is 1 dB.

```
10  PL=POLL(M,250,-6000,0,100)
20  PL=PL/100
30  PRINT "Poll Level=",PL,"dBm at",M
40  STOP
```

PLRH, PLLH, PLRL and PLLL functions

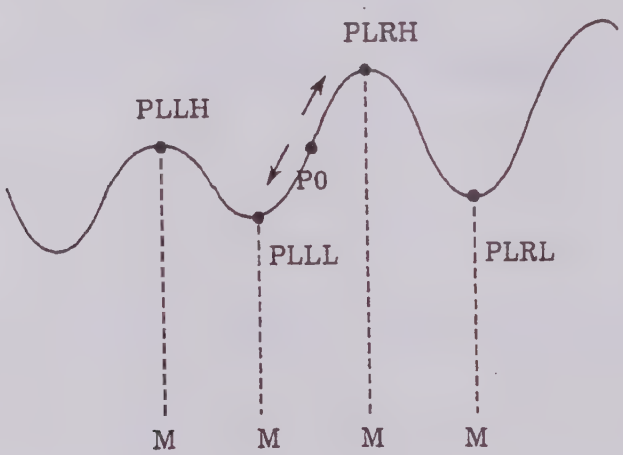
(1) Function

These functions obtain the first inflection top and bottom values and their measurement points by searching from a starting point in the specified memory.

(2) Format

```
PLRH(M,PØ,N[,R])
PLLH(M,PØ,N[,R])
PLRL(M,PØ,N[,R])
PLLL(M,PØ,N[,R])
```

- Resolution for difference between adjacent inflection top and bottom values
- Measurement memory specification
- Search starting point
- Measurement points of the inflection top and bottom values
- Inflection top and bottom values



Notes:

- If the difference between the adjacent inflection top and bottom values is smaller than R, the two points are not the inflection points. If R is omitted, it is assumed to be 0.
- If there is no inflection top and bottom point, M is assumed to be 0 at PLLH and PLLL and M is assumed to be 1001 at PLRH and PLRL; the measured value at point 0 is PLLH and PLLL and that at point 1001 is PLRH and PLRL.
- N specified by measured memory must be from 0 to 7, 8 or 9. (No real number memory can be used.)

(3) Program example: Obtains inflection top level searched right of the measurement point 200 in measurement memory TRACE-B, where resolution is 3 dB.

```
1Ø  PL=PLRH(M,25Ø,1,3ØØ)
2Ø  PL=PL/1ØØ
3Ø  PRINT "Peak Level=",PL,"dBm at",M
4Ø  STOP
```

PFRQ function

(1) Function

This function finds the frequency of the specified point or time in the memory.

(2) Format

```
PFRQ(PØ)
      └──── Specified point
```

Notes:

- When the effective trace setting on the CRT is frequency domain (TRACE-A, B, BG), the frequency is output; and when it is time domain (TRACE-TIME) the time is output.
- Frequency is output in 1 Hz units and time is output in 1μs units.
- This function finds frequency values by the following equations:

$$\text{Frequency}=\text{start frequency}+\frac{P0}{500}*(\text{frequency span})$$

(3) Program example:

Obtains maximum level between the measurement points 100 and 300 and frequency at that point in the measurement memory TRACE-A.

```
1Ø  GMAX=MAX(M,1ØØ,3ØØ,Ø)
2Ø  FR=PFRQ(M)
3Ø  GMAX=GMAX/1ØØ
4Ø  FR=FR/1E6
5Ø  PRINT "Peak Freq=",FR,"MHz"
6Ø  PRINT "Peak Level=",GMAX,"dBm"
7Ø  STOP
```

SUM function

(1) Function

This function finds the sum of the memory contents of a certain interval in the specified memory.

(2) Format

```
SUM(PØ,P1,N)
    |  |  |
    |  |  └── Memory specification
    |  └───── Area Specification (PØ<P1)
    └──────── Sum
```

$$SUM = \sum_{k=P0}^{P1} L(k)$$

(3) Program example: Obtains average value between the measurement points 240 and 260 (21 points) in measurement memory TRACE-A.

```
1Ø  S=SUM(24Ø,26Ø,Ø)
2Ø  AV=S/21/1ØØ
3Ø  PRINT "Average=",AV:F7.2,"dBm"
4Ø  STOP
```

Note: When the measurement memory contains invalid data (points with marker level displayed as ***), that data is assumed to be –30000 (=–300.00 dBm) and calculation is performed.

PSML and PSMH functions

(1) Function

This function finds the point where the sum equals or exceeds the specified value while adding the memory contents sequentially by searching from a starting point in the specified memory.
(For example, this is used to measure the occupied bandwidth)

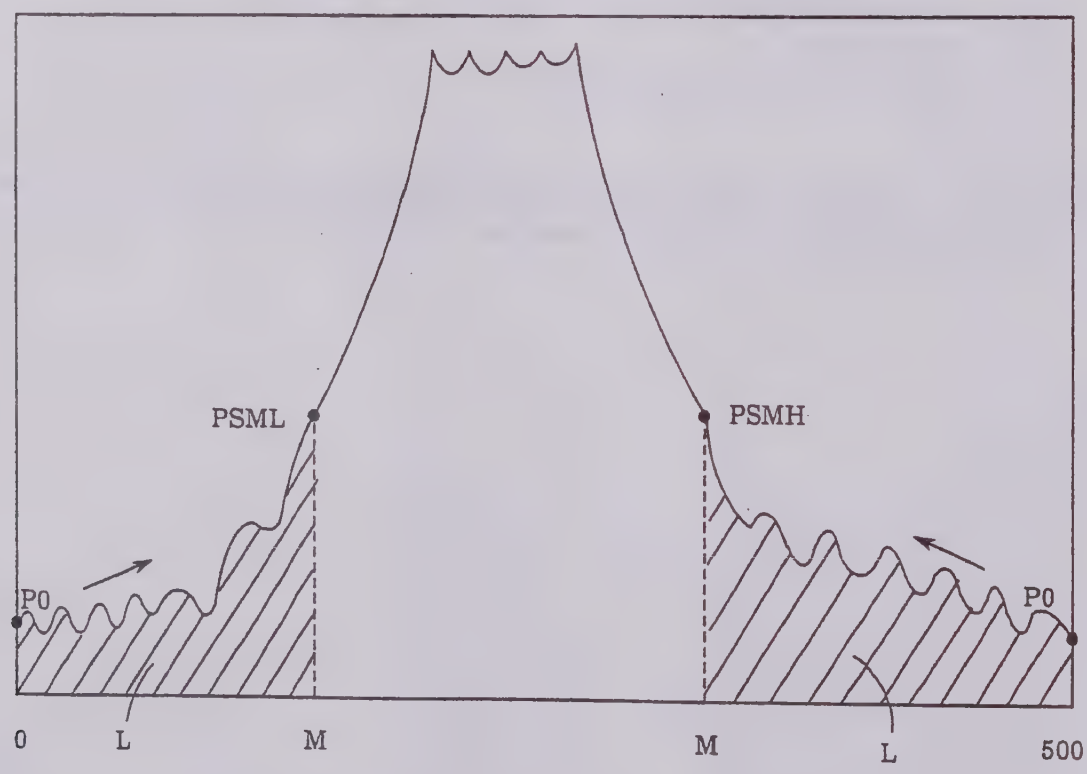
Finding method of the frequency or time depends on the specified waveform memory number.

See Section 5, "BNDL, BNDH, MESL and MESH functions" for details.

(2) Format

```
PSML(M,PØ,L,N)
PSMH(M,PØ,L,N)
```

- N — Memory specification
- L — Specified value
- PØ — Search starting point
- M — Measurement point
- M — Frequency at measurement point (or time)



PSML: Finds the minimum value of M that satisfies

$$L \leqq \sum_{k=P0}^{M} L(k)$$

PSMH: Finds the maximum value of M that satisfies

$$L \leqq \sum_{k=M}^{P0} L(k)$$

(3) Program example: Converts the measurement data in measurement memory TRACE-A to real value of mW unit, obtains sum of total data and frequency of the point, where sum equals 0.5% of the total sum adding the memory contents by searching from left end (address 0).

```
1Ø  CALL CONV(2,Ø,6,Ø,5ØØ)
2Ø  T=SUM(Ø,5ØØ,6)
3Ø  L=T*Ø.ØØ5
4Ø  FR=PSML(M,Ø,L,6)
5Ø  FR=FR/1E6
6Ø  PRINT "Point=",M
7Ø  PRINT "Freq=",FR,"MHz"
8Ø  STOP
```

DPOS and DNEG functions

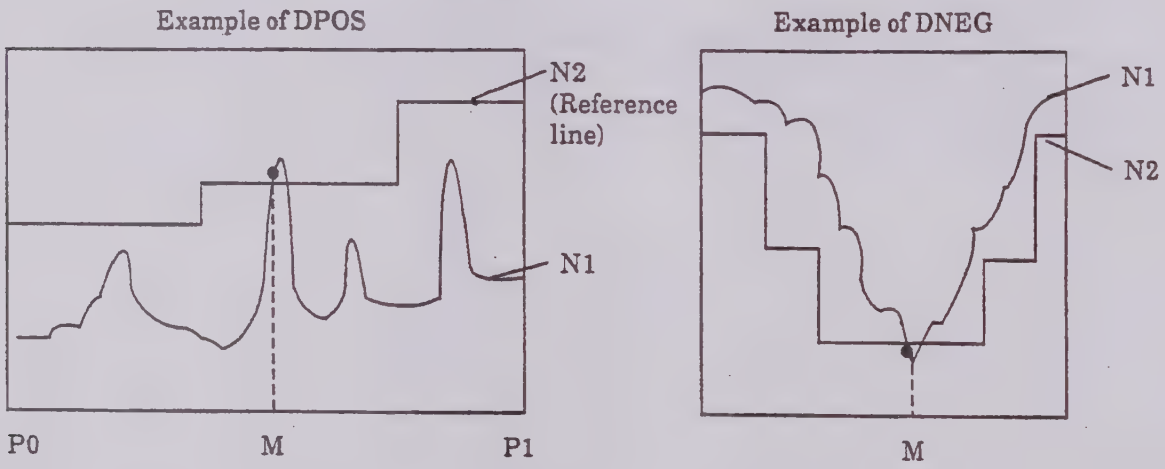
(1) Function

These functions compare the contents of two memories by address. If a value in one memory is larger (or smaller) that the other even if at only one point, the function value is assumed to be 1. Otherwise, 0 is output. (For example, this is used to judge GO/NOGO for the standard.)

(2) Format

```
DPOS(M,PØ,P1,N1,N2)
DNEG(M,PØ,P1,N1,N2)
```

- Memory specification
- Area specification
- Measurement point (the leftmost point exceeding the condition)
- Results Ø: GO
  1 : NOGO



DPOS
- 1 when there is a N1>N2 point
- 0 when there is no N1>N2 point

DNEG
- 1 when there is a N1<N2 point
- 0 when there is no N1<N2 point

(3) Program example: Compares the measurement data in measurement memory TRACE-A with measurement data in measurement memory TRACE-B and displays GO or NOGO.

```
1Ø  X=DPOS(M,Ø,5ØØ,Ø,1)
2Ø  IF X=Ø PRINT "GO"
3Ø  IF X=1 PRINT "NO GO"
4Ø  STOP
```

SECTION 6

REMOTE CONTROL COMMANDS USED WITH PTA PROGRAM/LIBRARY

TABLE OF CONTENTS

Outline 6-3

PTA Dedicated Remote Control Commands 6-4

SECTION 6
REMOTE CONTROL COMMANDS USED WITH PTA PROGRAM/LIBRARY

Outline

Remote control commands to control the main frame side, using PUT and WRITE 1000 texts in a PTA program/library, are sent. Also, using GET, COM and READ 1000 texts, measurement parameters and measurement results of the main frame side are read out. Remote control commands available here include all control and inquiry commands defined on the MS2665C/67C/68C main frame side. In addition, there are also remote control commands specially prepared for PTA programs/libraries.

PTA Dedicated Remote Control Commands

When setting or reading parameters of a measuring instrument on the PTA main frame side, messages in the remote control command format are sent using the WRITE 1000 or READ 1000 statement.

In PTA, besides the remote control commands of MS2665C/67C/68C, the following messages can be sent out.

Function		Message	
Port Switching	Control	`PORT␣1`	; Selects RS-232C as the PTA control port.
		`PORT␣2`	; Selects GPIB as the PTA control port.
		`PORT␣3`	; Selects the parallel (centronics) as the PTA controller port.
	Request	`PORT?`	; Requests the PTA control port.
Event Occurrence DELAY (Clock 1)	Control	`EDLY␣t`	; Sets the DELAY time an event interrupt will occur. DELAY time: 1 seconds up to 1 hour (in 1 s step)
Event Occurrence TIME (Clock 2)	Control	`ETIM␣t1,t2,t3`	; Sets the time an event interrupt will occur Seconds: Up to 59 seconds Minutes: Up to 59 minutes Hours: Up to 23 hours
Event Occurrence CYCLE (Clock 3)	Control	`ECYC␣t`	; Sets the cycles an event interrupt will occur. Cycle: 1 seconds up to 1 hour (in 0.1 s steps)

- For details on the WRITE 1000 and READ 1000 statements, see Section 4, "Setting measurement parameters (PUT and WRITE 1000 statements)" and "Measurement parameter/ data read (GET, COM and READ 1000 statements)".
- For details on event interrupts, see Section 4, "ENABLE EVENT statement".
- The control port (for the WRITE, READ, LISTG statements and other GPIB statements supported by the PTA) is the port selected by the PORT command except when these statements are executed with a direct port specification.
  In the initial state, the GPIB1 port is selected as the PTA control port.
- Ports specified by the port switching command are not initialized by PTA→OFF.

SECTION 7

EXTERNAL INTERFACE IN PTA

TABLE OF CONTENTS

Outline 7-3

Selection of Controlled Interface Port from PTA 7-4

RS-232C Functions in PTA 7-5

GPIB Functions in PTA 7-7

- Function as controller 7-7
- Function as device 7-11

Functions of Parallel (centronics) in PTA 7-12

Dual Port Memory 7-13

SECTION 7
EXTERNAL INTERFACE IN PTA

Outline

MS2665C/67C/68C provides an RS-232C interface and a GPIB interface as standard, and a parallel (centronics) interface (option 10) is optionally available. These external interfaces can be controlled from PTA.

Selection of Controlled Interface Port from PTA

An interface port controlled from PTA is selected by the "connection port for peripheral devices (Connect to Peripheral)" of the Interface menu.

(1) Press [SHIFT] + [.:Interface] keys.

(2) Press the F6 key "connection port for peripheral devices (Connect to Peripheral)" several times to display candidate interface ports for selection.

If the interface port to be controlled from PTA has been set as the "connection port for the external controller (Connect to Controller)" or the "connection port for the printer/plotter (Connect to Printer/Plotter)", first switch the selection to another port or make it "no connection (NONE)" and then operate the F6 key "connection port for peripheral devices (Connect to Peripheral)".

Also, using the PORT remote command or CALL IFC subroutine, it is possible to make the external interface port forcibly controllable from PTA.

- PORT_1: This command forcibly sets the connection port for external devices as the RS-232C interface.
- PORT_2: This command forcibly sets the connection port for external devices as the GPIB interface.
- PORT_3: This command forcibly sets the connection port for external devices as the parallel (centronics) interface.
- CALL IFC: This command forcibly sets the connection port for external devices as the GPIB interface.

RS-232C Functions in PTA

(1) Program listing

The LISTG command lists programs from the RS-232C port to an external printer.

(2) Data sending

The WRITE statement sends data to a device connected to the RS-232C.

```
WRITE␣M,Variable[:Format][,Variable[:Format]····]
```

Output data (A character constants available.)

External device address (numeric constant or variable used.)

(3) Data receiving

The READ statement receives data from a device connected to the RS-232C.

```
READ␣M,Variable [,Variable····]
```

Received data is input in the variable.

External device address (numeric constant or variable used.)

(4) Time-out

The time-out time is input as five seconds (initial value).

Use the following GPIB command to change the time-out time:

```
TOUT␣t
```

t=0 to 255 seconds (second unit)

If t=0 is specified, no time-out is set.

(5) Terminating Codes for READ/WRITE Statements

The following terminating codes are used for the RS-232C port.

Send terminators

<Port> command	Terminator code
WRITE LISTG	Either CR+LF or LF (Comply with TRM command)

Receive terminators

<Port> command	Terminator code
READ	LF or CR + LF

GPIB Functions in PTA

Function as controller

When the GPIB interface port is set as the "connection port for peripheral devices (Connect to Peripheral)", GPIB functions as a controller.

(1) Program listing

Lists programs to an external printer by using the LISTG command through the current GPIB port.

(2) IFC sending

Sends the "Interface Clear" to the device on the GPIB by using the CALL␣IFC statement.

(3) Controller right allocation

Allocates controller right to the device with the address specified by M by using the CALL␣TCT (M) statement.

(4) Data sending

Sends the data to the device on the GPIB by using the WRITE statement

```
WRITE␣M,Variable[:Format][,Variable[:Format]...]
```

Output data (A character constant is possible.)

Address of external device (A numeric constant or numeric variable is used.)

NOTES

When M is 1000, the functions of the MS2665C/67C/68C main frame are set. Also, this operations are performed in either the controller or device mode at this time.

(5) Data reception

Receives the data from the device on the GPIB by using the READ statement

```
READ␣M,Variable [,Variable...]
```

Received data is input in variable.

Address of external device (A numeric constant or numeric variable is used.)

NOTES

When the specified GPIB port is the device port, WRITE and READ statements access the dual-port memory.

NOTES

When one- or two-digit value (e.g.,5 or 17) is specified for an address, the value indicates the address of the device connected to the port specified by the PORT command of the GPIB command (Indirect Port Specification). When a three-digit value (e.g.,105 or 217) is specified, the high-order digit indicates the port number, and two low-order digits indicate the address of the device connected to the port indicated by the above port number. (Direct Port Specification).

The two lower digits of an address at indirect or direct port specification have no meaning in RS-232C. However, these digits should still be specified for form's sake.

Example:

`WRITE␣5,"ABC"`	Data is sent to address 5 through the current port (indirect port specification).
`WRITE␣1Ø5,"ABC"`	Data is sent to address 5 through the specified port No.1 (RS-232C) (direct port specification).
`READ␣217,A$`	Data is input from address 17 through the specified port No.2 (GPIB) (direct port specification).

These address specifications are effective for the WRITE, BWRITE, WWRITE, READ, BREAD, WREAD and LISTG statements.

The relationship (between the port specification command and controller port) is as follows:

	Indirect port specification	Direct port specification	
	WRITE 5	WRITE 1Ø5	WRITE 2Ø5
At power-on or after "PORT␣1" execution	*1 The RS-232C port is the controller port.	*1 The RS-232C port is the controller port.	*2 The GPIB port is the controller port.
After "PORT␣2" execution	*2 The GPIB port is the controller port.	*1 The RS-232C port is the controller port.	*2 The GPIB port is the controller port.

*1 Address specification in the RS-232C has no meaning. However, the address should still be specified for form's sake.

*2 If the GPIB port is not the controller port due to the CALL IFC statement, it controls the dual port memory. In this case, the LISTG statement becomes ineffective.

When the specified port is a device port, data is written to and read from the dual port memory. In this case, the BWRITE, WWRITE, BREAD, WREAD, and LISTG statements cannot be used.

(6) Time out

The time-out value is 30 s (initial value).

The following GPIB command is used for change of thime-out value.

GTOUT␣t t=0 to 225 s (in 1 s steps)

When t=0 is specified, no time-out is set.

(7) Terminating Codes for READ/WRITE Statements

The following terminating codes are used for the GPIB ports.

Talker (send) terminators

<Port> command	Terminator code
<GPIB> WRITE LISTG	Depends on TRM command. eiter CR + LF or LF

Listener (receive) terminators

<Port> command	Terminator code
<GPIB> READ	LF or CR + LF

Note:
The TRM command shown below is a GPIB command.

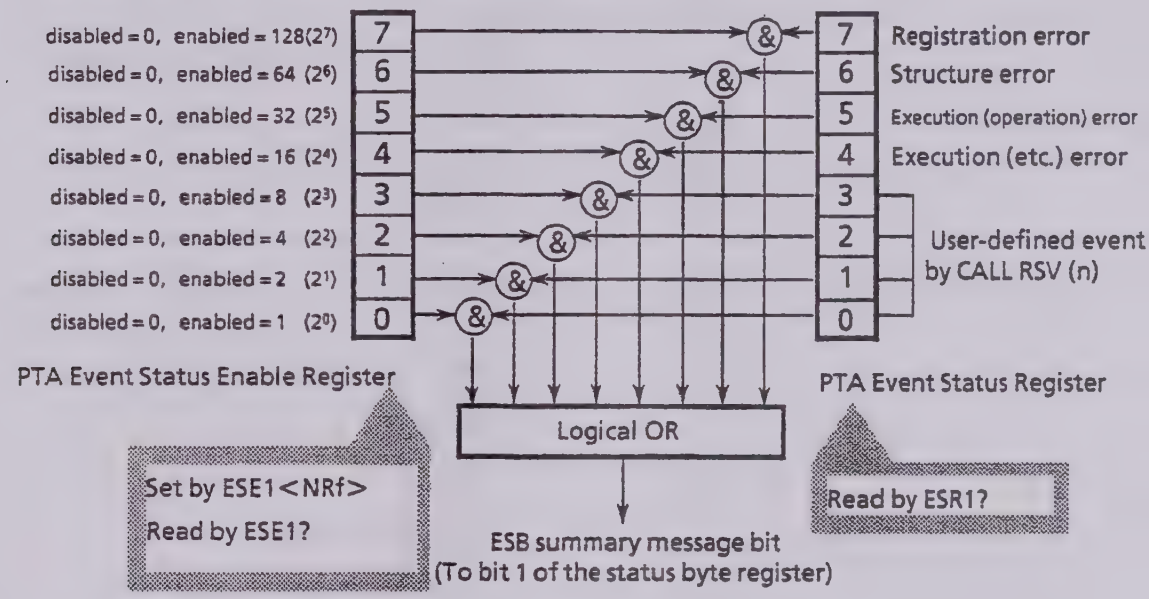
TRM␣1 (CR + LF)
TRM␣Ø (LF only)
Initial value : LF only

Function as device

When the GPIB interface port is set as the "connection port for the external controller (Connect to Controller)", GPIB functions as a device.

(1) Service request sending

Sends a service request command to an external controller by using the CALL_RSV (M) statement.



Bit	Event name	Description
7	Registration error	Error at program registration
6	Structure error	Error on program structure
5	Execution (operation) error	Error at operation on program execution
4	Execution (etc.) error	Error at other than program operation
3	(User-defined event)	(User defined by CALL RSV (n))
2	(User-defined event)	(User defined by CALL RSV (n))
1	(User-defined event)	(User defined by CALL REV (n))
0	(User-defined event)	(User defined by CALL RSV (n))

Functions of Parallel (centronics) in PTA

(1) Program listing

The LISTG command lists programs from the parallel (centronics) port to the external printer.

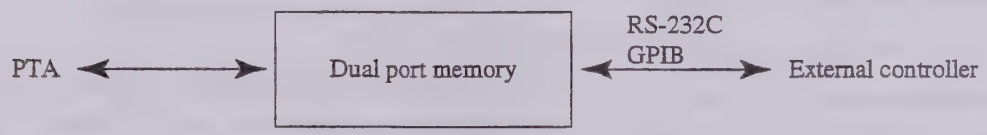
Dual Port Memory

(1) Application and configuration

The dual port memory is built in PTA, and data can be freely written and read from PTA and the external controller.

Data and measurement results obtained in the PTA program/library are outputted to the external controller through this memory, and used for performing communication between PTA and external controller.

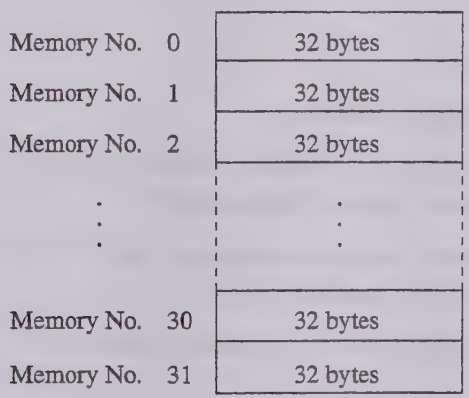
The external controller writes to and reads from the dual port memory through the interface set as the "connection port for the external controller (Connect to Controller)".



The dual port memory consists of thirty-two 32-byte memories. The memories are accessed by specifying the memory number.

Memory numbers from 0 to 31 can be specified.

Dual port memory configuration



(2) Writing data to dual port memory

Format

- Writing from PTA

```
WDPM memory number, write data   or
PUT(or WRITE 1ØØØ) " PMY memory number, write data"
```

- Writing from external controller

```
" PMY memory number, write data"
```

- When writing data to the dual port memory, be sure to specify the memory number. Data is written sequentially, beginning from the first byte of the specified memory number.
- A 1-byte termination code (LF) is added at the end of the write data.
- When the write data size exceeds 32 bytes, it can be written to the next memory. When the write data size is exactly 32 bytes, the termination code is stored at the beginning of the next memory. However, when data has been written up to the last byte of the last memory number, the termination code is not added.
- When writing past the last byte of the last memory number is attempted, an error is generated and writing is not performed. In this case, the previously written data is retained.
- Data is always stored in memory as ASCII data. When data is written from the PTA, its storage size differs as follows, depending on the type of data:

1) Character constant/variable

- Written as 1 byte/1 character ASCII data.
- When unformatted character variable data is written, (number of bytes of array size)+(1 byte: space code) is written. The termination code is written at the end.
- When upper case formatted character variables are used, a 1-byte space code is written at the end of the data. The termination code is written at the end.
- When character variables are used, the number of characters in " " are written. The termination code is written at the end.

2) Numeric variable

- Numerics are converted to character strings (ASCII data) and data of that size is written.
  The minus sign and decimal point require one byte each.
  The termination code is written last.

3) Bit variable

- The 0/1 numeric of each bit is converted to a character string (ASCII data) and data of that size is written as 1 byte/1 bit.
- The storage format when the data is formatted/unformatted is the same as when character variables are used.
- The BWRITE and WWRITE statements cannot be used.

Examples:

- Writing from PTA

```
WDPM Ø, "MEASEND"        : Write "MEASEND" to Memory No. 0.
```

- Writing from external controller

```
"PMY Ø, MEASSTART"       : Write "MEASSTART" to memory No.0.
```

Notes:

- The WDPM statement is a dedicated statement for writing data to dual port memory.
- The PUT or WRITE 1000 statement is mainly used to set measurement parameters of the main frame. However, messages in the same format as setting from the external controller can be written using these commands by sending messages in the remote control command format from PTA.

(3) Reading data from dual port memory

Format

- Reading from PTA

```
RDPM memory number, input variable[,input variable..]  or
PUT(or WRITE 1ØØØ) "PMY? read start memory number, number of memories"
+READ 1ØØØ, input variable[,input variable]
```

- Reading from external controller

```
"PMY? read start memory number, number of memories" + read command
```

- When reading data from the dual port memory, be sure to specify the memory number. Everything up to the termination code (LF) is, as a rule, output as one data item.

  However, when dual port memory was read up to the last byte of the last memory number, the data is assumed to end at that point.

- When data was written over multiple memories and is read by specifying an intermediate memory number, the intermediate data is read.

- As a rule, when data is read from the PTA, the data up to the termination code is read. However, if the data contains commas (" , "), the commas are assumed to be delimiters and the data up to the front of the comma is stored in the input variable. Therefore, in this case, multiple input variables must be specified.

  When the number of delimited data and the number of input variables is different, a write error (when the number of input variables is large) may be generated, or the output data may remain inside (when the number of input variables is small).

  To avoid a comma being considered a data delimiter, store the data up to the termination code in one input variable by specifying " ; " at the end of the statement.

  In this case, only one input variable can be specified.

- When data is read from an external controller and when data is read from the PTA with the PUT or WRITE 1000 statement, use the "PMY?" command. The "PMY?" command can specify the read start memory number and the number of memories to be read. In this case, the data from the beginning to the termination code of each memory number is delimited into the specified number of memories by commas and is output.

- When the data in the dual port memory is assigned to input variables, it may not be possible to assign the data to an input variable type different from the assignment data. In this case, a read error is generated.

- The BREAD and WREAD statements cannot be used.

Examples:

- Reading from PTA

  `RDPM Ø, A$` : Read data from Memory No. 0 and store it in character variable A$.

- Reading from external controller

  `"PMY? Ø, 3"` : Issue a memory data output request for Nos. 0 to 3 (memory Nos. 0, 1, 2).

Notes:

- The RDPM is a dedicated statement for reading data from dual port memory.

(4) Details of write/read the dual-port memory

Control command from external controller

```
"PMY␣Ø,ABC"
"PMY␣1,123"
"PMY␣2,XYZ"
```

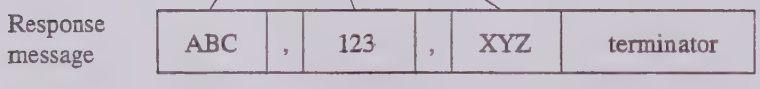
Contents of dual-port Memory

Memory 0	ABC (LF)
Memory 1	123 (LF)
Memory 2	XYZ (LF)

After executing statements shown on the above left, the contents of the dual-port memory are as shown on the above right.

When these data are read using "PMY?" command, the following contents are stored in variables A$, B$, and C$, respectively.

```
PUT␣"PMY?␣Ø,3"
READ␣1ØØØ,A$,B$,C$
```



- Comma <,> in dual-port memory

  The output data for PMY? is assumed to be everything from the beginning to the <termination> code of the specified memory number. The output data includes the memory contents up to (but not including) the terminator. If a comma <,> is included in the contents, it indicates the presence of output data.

  In contrast, data in the READ statements for the PTA and controller are separated by commas and sequentially assigned to data variables. Therefore, the number of output variables generated by the PMY? command may be different from the number of variables required for the corresponding statement.

Contents of dual-port Memory

Memory 0	ABC, DEF (LF)
Memory 1	XYZ (LF)

Execute the statements shown below to read the contents of the dual-port memory at addresses 0 and 1.

```
PUT␣"PMY?␣Ø,2"
READ␣1ØØØ,A$,B$
```

The ABC represents data for variable A$ and the DEF represents data for variable B$. The contents of the memory 0 are separated by a comma (,). This comma separates the data into two data values. Consequently, the XYZ data in the memory 1 is not read. Therefore, the number of input variables in the READ statement must be set to three.

SECTION 8

PTA ERROR MESSAGES

TABLE OF CONTENTS

Error Message Format 8-4

ERROR Statement 8-5

ERRMAIN Statement 8-6

Error Processing Subroutines 8-7

- ON ERROR statement 8-7
- OFF ERROR statement 8-7
- Returning from error processing subroutines (RETERR, RETRY, RESUME and GIVE UP statements) 8-7
- ERRREAD (m) function 8-8

Error List 8-9

SECTION 8
PTA ERROR MESSAGES

An error message is displayed when an error is detected in the PTA command or program.

There are two types of errors;an execution-stop error (fatal: F) and an execution-continuable error (warning: W).

- Execution-stop error (F:Fatal) :
  This type of error stops the execution of the program unconditionally.
- Execution-continuable error (W:Warning error) :
  When there is no ERROR statement in the line next to the line where this type of error occurs, the execution stops; but if there is an ERROR statement, execution continues.
  And also, error interruption process can continue the execution.

Error Message Format

The error message is displayed in the following format.

- PTA program:

```
ERROR Error level Error number[,Error-occurrence line number]
```

This is displayed at the program execution.

- PTA library:

```
ERROR Error level Error No.[,erred line No.,erred program name]
```

This is displayed at program execution.

No.300 and on are errors of the library program itself.

ERROR Statement

(1) Function

For an execution-continuable error generated at program execution, execution can be continued by using the ERROR statement.

An ERROR statement can be programed over several lines.

(2) Format

```
ERROR(21Ø,1ØØØ)
```

Next program line to be executed

Error number

This statement means that when the error (generated in the previous line) corresponds to the error number 210,the program of line 1000 is executed.

When it does not correspond, the error message is displayed and execution stops.

(3) Example

```
1Ø X = Ø
2Ø Y = 1ØØ/X
3Ø ERROR(21Ø,1ØØ)   ; If the error (210: the divisor is 0) occurs, jump to line 100.
4Ø Y = Y+5Ø
 :
```

ERRMAIN Statement

(1) Function

To branch to the main routine whenever a execution continuable ERROR occurred, use the ERRMAIN statement.

(2) Format

```
ERRMAIN (error number)
```

(3) Example

```
1Ø   INPUT A
2Ø   GOSUB 1ØØØ
3Ø   :
     :
1ØØØ WRITE 217,A
1Ø1Ø ERRMAIN(222)      ; If the error (222) occurs because the data of WRITE statement can
1Ø2Ø                     not output, the program returns to the main routine.
     :
```

Note: If the ERRMAIN statement has been executed in the highest level of the routine, ERROR 213 is generated.

Error Processing Subroutines

ON ERROR statement

(1) Function

Registers the subroutine to branch (interrupt) to when an error occurs.

(2) Format

```
ON ERROR line number(or *label)
```

After executing this statement and an error that is possible to continue execution occurs, an interrupt occurs and the error processing subroutine is executed from the line number (or label) specified.

OFF ERROR statement

(1) Function

Releases the registered subroutine to branch (interrupt) to when an error occurs.

(2) Format

```
OFF ERROR
```

After executing this statement, error interrupts will not occur.

Returning from error processing subroutines (RETERR, RETRY, RESUME and GIVE UP statements)

(1) Function

Returns from an error interrupt.

(2) Format

`RETERR`	(Continues from the statement following the statement where the error occurred.)
`RETRY`	(Continues reexecuting from the statement caused the error.)
`RESUME`	(Continues from specified line.)
`GIVEUP`	(Stops program execution.)

Note: See Section 4, "RETERR statement" to "GIVEUP statement".

ERRREAD (m) function

(1) Function

Reads the line where the error occurred and the error code in the middle of an error processing subroutine.

(2) Format

V=ERRREAD(Ø)	(Error code)
V=ERRREAD(1)	(Line where error occurred)

(3) Example

```
1ØØ ON ERROR 2ØØ              ; Jumps to line 200 on error
11Ø INPUT X
12Ø Y=1ØØ/X
13Ø PRINT Y
14Ø GOTO 11Ø
15Ø STOP
2ØØ C=ERRREAD(Ø)
21Ø IF C=21Ø GOSUB 3ØØ        ; For "Divide by zero", continues to execute from line 130
                                displaying "ERROR/0".
22Ø IF C<>21Ø GIVEUP          ; On other errors, stops program execution
23Ø RETERR
3ØØ PRINT "ERROR /Ø"
31Ø RETURN
```

Error List

Table 8-1 shows the error number and error cause. In the table, F (Fatal) denotes the execution-stop error and W (Warning) denotes the execution-continuable error.

Table 8-1 PTA Error List

Error No.	Cause of error	W,* F**
0	[↵] key pressed but no commands or statement input	F
1	Number of characters (representing variable) exceeds 8, or number of characters (representing program name) exceeds 6.	W
2	Format of numeric constant in correct Example : `Ø..1 4.5EE2`	W
3	Too many input digits, or value of numeric constant too large or too small (Format of numeric constant incorrect)	W
4	Format of character string constant incorrect Example : `A$="ABC`	W
5	Format incorrect Example : `PRINT A:G6.2`	W
6	Statement cannot be interpreted (command format error) Example : `GOTO ABC`	W
7	Statement insufficiently described Example : `GOTO`	W
8	Statement excessively described Example : `GOTO 1ØØ,2ØØ`	W
9	Number of variables exceeds 256 (Up to 256 user-defined variables can be written)	W
10	Character cannot be interpreted Example : `-1ØØ`	W
11	Format (of binary or hexadecimal constant) incorrect Example : `8#=# 11Ø`	W
12	Value (of binary or hexadecimal constant) too large Binary constant : up to 8 characters Hexadecimal constant : up to 2 characters Example : `8#=#1ØØØØØØØ`	W
13	Number of format digits too large Example : `PRINT A:F6.5`	W

*W : Execution-continuable error (Warning)
**F : Executiong-stop error (Fatal error)

Table 8-1 PTA Error List (Continued)

Error No.	Cause of error	W, F
14	Command operand cannot be interpreted Example : `LIST A,B`	W
15	Command operand insufficient Example : `LISTG`	W
16	Command operand excessive Example : `DELETE 1Ø,1ØØ,3ØØ`	W
17	Line number exceeds 65535 (Program line number is 1 to 65535)	W
20	Program on a line too long to assemble	W
21	Undefined-line-number label used as command operand	W

Note : Errors 0 to 21 may occur during program input or command execution.
Errors 6 to 8, however, may also occur during statement execution.

Table 8-1 PTA Error List (Continued)

Error No.	Cause of error	W, F
101	Value of command operands 1 and 2 incorrect Example : `LIST 1ØØ,1Ø`	F
102	Program exceeds memory capacity	F
103	No Line number or program, designated by command (LIST, LISTG, DELETE, RENUM, and SAVE commands)	F
104	Since number of GOTO or GOSUB statements excessive (>100), RENUM statement cannot be executed	F
105	Since line number (specified by GOTO or GOSUB operand) not found, RENUM statement cannot be executed	F
111	Line number exceeds 65535 when RENUM and PCOPY statements executed	F

Note : Errors 101 to 105 and 111 may occur during command execution.

Table 8-1 PTA Error List (Continued)

Error No.	Cause of error	W, F
120	Media write-protected	W
121	Media not installed	W
122	Media memory overflow	W
123	Specified program not stored in media	W
124	Media faulty	W
125	Memory type incorrect	W
126	Media formatting incorrect	W
127	Media not formatted	W
150	Label is not defined or defined more than once	F
151	No DATA statement	F
180	Error of the command transmitted from PTA to main frame	W

Note : Errors 120 to 127 may occur when a command or statement attempts to access the media (PMC or FD).

Table 8-1 PTA Error List (Continued)

Error No.	Cause of error	W, F
201	Program cannot be resumed (CONT command)	F
202	Specified line number missing RUN command executed without program (RUN, CONT commands and GOTO, GOSUB statements)	W
203	Array subscript (in DIM statement) incorrect (The array subscript must be from 1 to 1024; the bit array subscript must be from 1 to 8, and the character array subscript must be from 1 to 255.)	W
204	Used as simple, or system variables before array declaration by DIM statement	W
205	Array declaration overlapped	W
206	Insufficient variable memory capacity due to program memory overflow	F
207	Arithmetic operation of character data or bit data	W
208	Data-type combination incorrect for conversion	W
209	Overflow or underflow occurred	W
210	Divide by 0	W
211	Value of arithmetic function parameter too large or too small	W
212	Nesting (by subroutine, FOR and NEXT statement) exceeded 10 levels	F
213	No return destination specified for RETURN statement	F
214	Comparison cannot be made by IF statement Right and left side data-type combination incorrect	W

Table 8-1 PTA Error List (Continued)

Error No.	Cause of error	W, F
215	SOS statement is executed	F
216	No corresponding FOR statement. That is, there are excess NEXT statements. (RUN, CONT command and GOTO, GOSUB statements)	W
217	Input data format (in INPUT statement) incorrect	W
218	Input data (in INPUT statement) insufficient	W
219	Excess amount or too large input data in INPUT statement	W
220	Minus sign used in exponentiation Example : `-1!5`	W
221	Data can not be input in GPIB (Talker device not connected)	W
222	Data cannot be output in GPIB	W
223	Parameter (in the statement) outside range or variable type incorrect Example : `WAIT A$`	W
224	Simple variable includes array subscript	W
225	Array variable has no subscript	W
226	Array-variable subscript out of boundary Note that the subscript range declared in DIM J(5) is J(0) to (4).	W
227	GPIB execution is impossible because the PTA is set as the device	W
228	GPIB execution is impossible because the PTA is set as the controller	W

Table 8-1 PTA Error List (Continued)

Error No.	Cause of error	W, F
229	STOP statement (to terminate program execution) not specified	W
230	Attempt made to refer to non-referable system variable	W
231	Attempt made to assign non-assignable system variable	W
232	Array variable subscript not numeric	F
233	Parameter (in boolean function) not bit type	W
234	Parameter of FOR statement is character or bit type	W
235	The I/O type specification in the EVENT statement is out of range (0 to 99).	W
236	Variable of NEXT statement does not correspond to that of FOR statement specified before NEXT statement Example : `3Ø FOR C=...` `9Ø NEXT D`	W
237	Six or more character constants and variables used in INPUT, PRINT, READ or WRITE statement Example : `PRINT"FREQ",F(C),"Hz","LEVEL",LEV,"dBm"`	W
238	Variable type and format type of PRINT or WRITE statement do not agree	W
239	Operand (in LISTG, WRITE or READ statement) outside range (0 to 31) Example : `LISTG 35`	W
240	Variable or constant values of CALL statement or system function outside range	W
241	Vairable or constant type of CALL statement or system function incorrect	W
242	System variable used in CALL statement or system function	W

Table 8-1 PTA Error List (Continued)

Error No.	Cause of error	W, F
243	The RETURN or RETMAIN statement was used to return from event or error interrupt processing.	F
244	Media data file not open	W
245	Media data file opened	W
246	Media data already read	W
247	Media data type and variable type combination incorrect (unconvertible)	W
248	Excess amount or too large input data value in READ statement.	W
249	Insufficient input data in READ statement	W
250	Input data format (in READ statement) incorrect	W
251	The RETINT statement was used for something other than event interrupt processing. Or, the GOSUB statement was executed in the middle of event interrupt processing and the RETURN statement to return was not executed, but the RETINT statement was instead.	F
252	The RETERR, RETRY, RESUME, GIVEUP statements were used for something other than error interrupt processing. Or, the GOSUB statement was executed in the middle of error interrupt processing. Or, the RETURN statement to return was not used and one of the above statements was executed.	F
253	The ERRREAD function was executed for something other than error interrupt processing.	F
254	The STATUS function was executed for something other than event interrupt processing.	F

Table 8-1 PTA Error List (Continued)

Error No.	Cause of error	W, F
301	Library/program is being selected.	W
302	The specified measuring instrument library does not exist in the memory.	W
303	A program having the new program name specified by RENAME exists.	F
304	The file containing the same name as that of the program in execution was loaded.	W
305	The number of nesting by CALLIB has exceeded 10.	F
306	The library was executed during sequence registering/downloading.	F
307	The specified measuring instrument library is being executed.	W
308	The specified measuring instrument library is being locked.	W
309	Result of processing by the main frame's measuring instrument is abnormal.	W
310	The library is being registered.	W
311	The LIBRARY statement cannot be edited.	W
312	CHKFILE was executed to the .MNU file.	W
313	The specified measuring instrument library resides in ROM.	W
314	The COMCLEAR statement cannot be executed in the nested PTA library.	W

Anritsu Service and Sales offices

America

Anritsu Company (ACUS)

685-A Jarvis Drive Morgan Hill, CA
95037-2809, U.S.A.
Phone: +1-408-776-8300
Fax: +1-408-776-1738

Anritsu Company (ACUS-NJ) Dist.5

10 New Maple Avenue, P.O. Box 836
Pine Brook, NJ 07058-0836, U.S.A.
Phone: +1-973-227-8999
Fax: +1-973-575-0092

Anritsu Company (ACUS-Maryland) Dist.6

19630 Club House Road, #710
Gaithersburg, MD20879, U.S.A.
Phone: +1-301-590-0300
Fax: +1-301-216-2893

Anritsu Company (ACUS-Tx) NARO

1155 East Collins Blvd
Richardson, TX 75081, U.S.A.
Phone: +1-972-644-1777
Fax: +1-972-644-3416

Anritsu Electronics Ltd (ACCA)

5A-245 Matheson Blvd. East, Mississauga,
Ontario, L4Z 3C9, Canada
Phone: +1-905-890-7799
Fax: +1-905-890-2290

Anritsu Electronics Ltd (ACCA)

102-215 Stafford Road West Nepean,
Ontario, K2H 9C1, Canada
Phone: +1-613-828-4090
Fax: +1-613-828-5400

Anritsu Electronics Ltd (ACCA)

200-1405 Trans Canada Highway
Dorval, Quebec H9P 2V9, Canada
Phone: +1-514-421-3737
Fax: +1-514-685-9839

Anritsu Electronics Ltd (ACCA-Calgary)

Deerfoot Atrium, Suite 129, 6715-8th Street
N.E., Calgary, AB, T2E 7H7, Canada
Phone: +1-403-275-9855
Fax: +1-403-275-3609

Anritsu Eletrônica Ltd (ACBR)

Praia de Botafogo, 440-Sala 2401
CEP 22250-040, Rio de Janeiro, RJ, Brasil
Phone: +55-21-5276922
Fax: +55-21-537-1456

Anritsu Eletrônica Ltd (ACBR) Sao Paulo Branch Office

Praca Amadeu Amaral 27,
Primera Andar, Conj. 11, 12, 13, 14
Liberdade, Sao Paulo, Estado de
Sao Paulo, CEP : 01327-010, Brasil
Phone: +55-11-283-2511
Fax: +55-11-288-6940

Data Lab S.R.L

Edif. Ayfra, Pdte. Franco y Ayolas
Asuncion, Paraguay
Phone: +595-21-443-046
Fax: +595-21-441+935

Electro-Impex S.A.

P.O. Box 620-1000, San Jose, Costa Rica
Phone: +506-2-31-5701
Fax: +506-2-31-6531

Electronica 2000, S.A. DE C.V.

Blvd. Adolfo Lopez Mateos No.2016, Col.
Tlacopac, Del. Lopez Mateos, 01010,
Mexico D.F.
Phone: +525-662-8800
Fax: +525-662-5862

Electronic Engineering S.A.

Carretera de Circunvalavion, Sabanilla, Av
Novena, San Jose, Costa Rica
Phone: +506-2-25-8793
Fax: +506-2-25-1286

HDM Elquitecnica Cia. Ltda., Equitronics S.A.

Av. Republica de El Salvador, No.880,
Edificio, Almirante Colon 4to piso,
Quito, Ecuador
Phone: +593-9-704890
Fax: +593-2-48-2627

KRM Ingenieria Saciafs.

Viamonte 377,7° Piso 1053 Buenos Aires,
Argentina
Phone: +54-1-311-4165
Fax: +54-1-311-2297

SI Ltda.

E. Conchy Y Toro 65, Casilla 51888,
Santiago, Chile
Phone: +56-2-696-7534
Fax: +56-2-696-9665

Radiocom S.A.

Carrera 21 No.85-71, Conmutador
6100077, Santafe de Bogota, Columbia
Phone: +57-1-218-2054
Fax: +57-1-610-3272

Radiocomunicaciones cruz C.A.

Avda La Colina QTA. Elison, URB Colina
De Los Caobos, Caracas, Venezuela
Phone: +58-2-793-2322
Fax: +58-2-793-3429

Sakata Ingenieros S.A.

Av. Canaval Moreyra 840, Lima 27, Peru
Phone: +51-1225-7555
Fax: +51-1224-8148

Suministros Industriales S.A.

Casa 102, Calle 68 Este,
San Francisco, Balboa, Panama
Phone: +507-270-2328
Fax: +507-270-2329

Cabonorte S.A.

Colonia 1900, ESC. 603,
Montevideo, Uruguay
Phone: +598-2-430522
Fax: +598-2-418594

Europe

Anritsu Ltd (ACUK)

200 Capability Green, Luton
Bedfordshire, LU1 3LU, United Kingdom
Phone: +44-1582-433200
Fax: +44-1582-731303

Anritsu Ltd (ACUK-Manchester)

Kansas Avenue, Langworthy Park Salford,
Manchester M5 2GL, United Kingdom
Phone: +44-161-873-8041
Fax: +44-161-873-8040

Anritsu Ltd (ACUK-Bristol)

1230 Aztec West, Almondsbury
Bristol BS12 4SG, United Kingdom
Phone: +44-1454-615252
Fax: +44-1454-618017

Anritsu Ltd (ACUK-Livingston)

Unit 1, Knightsridge Industrial Estate
Turnbull Way, Knightsridge Livingston
EH54 8RB, United Kingdom
Phone: +44-1506-436111
Fax: +44-1506-436112

Anritsu GmbH (ACDE)

Grafenberger Allee 54-56
D-40237 Düsseldorf 1, Germany
Phone: +49-211-96855-0
Fax: +49-211-96855-55

Anritsu GmbH (ACDE-Sales Center South)

An der Steinernen Brücke 1 D-85757
Karlsfeld, Germany
Phone: +49-8131-3825-0
Fax: +49-8131-3825-95

Anritsu S.A. (ACFR)

9, Avenue du Québec ZA de
Courtaboeuf 91951 Les Ulis Cedex,
France
Phone: +33-1-60-92-15-50
Fax: +33-1-64-46-10-65

Anritsu S.A. (ACFR) (Toulouse Office)

Bureau de Toulouse
Région Centre Sud Ouest
Phone: +33-5-62070484
Fax: +33-5-62070668

Anritsu S.A. (ACFR) (Toulon Office)

Bureau de Toulon
Région Centre Sude Est
Phone: +33-4-94040264
Fax: +33-4-94040265

Anritsu S.A. (ACFR) (Rennes Office)

Bureau de Rennes
Région Ouest
Phone: +33-2-99521214
Fax: +33-2-99521224

Anritsu S.p.A. (ACIT)

Via Elio Vittorini, 129
00144 Roma EUR, Italy
Phone: +39-06-509-9711
Fax: +39-06-502-2425

Anritsu S.p.A (ACIT-Milano)

C.D. Colleoni, Via Paracelso, 420041
AGRATE B.ZA(MI), Italy
Phone: +39-039-65-7021
Fax: +39-039-605-6396

Anritsu AB (ACSE)

BOTVID CENTER
145 84 STOCKHOLM, Sweden
Phone: +46-8-53470700
Fax: +46-8-53470730

Anritsu AB-Norway Branch Office (ACNO)

Leangbukta 40 1370 Asker. Norway
Phone: +47-66-901190
Fax: +47-66-901212

Anritsu AB-Finland Branch Office (ACFI)

Piispanportti 9 FIN-02240 Espoo, Finland
Phone: +358-9-435-522-0
Fax: +358-9-435-522-50

Anritsu AB-Denmark Branch Office (ACDK)

SOHOJ 11, DK-2690 KARLSLUNDE
Denmark
Phone: +45-46160330
Fax: +45-46155299

C.N. Rood B.V.

Cort van der Linddenstraat 11-13, 2288
EV Rijswijk ZH, The Netherlands
Phone: +31-70-3996360
Fax: +31-70-3905740

C.N. Rood SA/NV

Pontbeeklaan 45, 1731 Zellik, Belgium
Phone: +32-2-4668199
Fax: +32-2-4662500

ELSINCO GMBH

h.e. Strelbishte, str. Kotlenski Prohod, bl.
96/6/14, BG-1408 Sofia, Bulgaria
Phone: +359-2-58-61-31
Fax: +359-2-58-16-98

ELSINCO Praha Spol.

Novedvorska 994, CZ 142 21 Praha
4-Branik , Czecho Republic
Phone: +42-2-49-66-89
Fax: +42-2-49-54-83

ELSINCO Budapest KFT

Pannonia utca 8. IV/I.
H-1136 Budapest, Hungary
Phone: +36-1-269-18-50
Fax: +36-1-132-69-27

ELSINCO Polska Sp. Z.O.O

ul. Dziennikarska 6/1, PL 01 605
Warszawa, Poland
Phone: +48-22-39-69-79
Fax: +48-22-39-44-42

ELSINCO Bratislava Spol. s.r.o.

Kudlakova 4, SK 844 15 Bratislava,
Slovakia
Phone: +42-7-784-165
Fax: +42-7-784-454

G'Amungason Co.

Skulagata 40, 101 Reykjavik, Iceland
Phone: +354-1-677887
Fax: +354-1-625045

GMP S.A.

Av. des Baumettes 19, CH-1020 Renens
1 Lausanne, Switzerland
Phone: +41-21-6348181
Fax: +41-21-6353295

Instrutek Oeriferi A/S

Christiansholmsgade DK-8700 Horsens,
Denmark
Phone: +45-75-611100
Fax: +45-75-615658

Kostas Karayannis SA

58, Kapodistriou str., GR-142 35 Nea Ionia,
Athens, Greece
Phone: +30-1-680-0460-4
Fax: +30-1-685-3522

Omnitecnica S.A.

Estrada Alfragide 2700 Amadora, Portugal
Phone: +351-1-471-55-17
Fax: +351-1-471-36-10

Pema Ltd.

Doromiskin, Dundalk, Co. Louth, Ireland
Phone: +353-42-72899
Fax: +353-42-72376

Unitronics S.A.

Plaza Espana 18, Torre de Madrid,
Pl. 12-ofc. 9, 28019 Madrid, Spain
Phone: +34-1-5425204
Fax: +34-1-5591957

Wien CHALL GMBH

Krichbaumgasse 25, 1120 Wien, Austria
Phone: +43-1-811-55140
Fax: +43-1-811-55180

Asia, Pacific and Africa

Anritsu Private Ltd (ACSG)

6, New Industrial Rd., #06-01/02
Hoe Huat Industrial Building
Singapore 536199
Phone: 65-282-2400
Fax: 65-282-2533

Anritsu Company Ltd (ACHK)

Suite 719, 7/F., Chinachem Golden Plaza,
77 Mody Road, Tsimshatsui
East, Kowloon, Hong Kong
Phone: +852-2301-4980
Fax: +852-2301-3545

Anritsu Company Incorporated (ACTW)

6F, 96, Sec. 3, Chien Kou North Rd.
Taipei, Taiwan
Phone: +886-2-2515-6050
Fax: +886-2-2509-5519

Anritsu Corporation, Ltd (ACKR) Head Office

14F Hyunjuk Bldg. 832-41,
Yeoksam-dong, Kangnam-ku,
Seoul, Korea
Phone: +82-2-553-6603
Fax: +82-2-553-6604~5

Anritsu Proprietary Ltd (ACAU)

Unit 3, 170, Forster Rd., Mt. Waverley
Victoria 3149, Australia
Phone: +61-3-9558-8177
Fax: +61-3-9558-8255

Anritsu Proprietary Ltd (ACAU)

Suite 304/2 Rowe Street Eastwood
NSW 2122, Australia
Phone: +61-2-9874-9044
Fax: +61-2-9874-9920

Anritsu Corporation Beijing Liaison Office

Room No.1515, Beijing Fortune Bldg. 5
Dong-San-Huan-Bei-Lu Chao-Yang
District Beijing 100004, P.R. China
Phone: +86-10-6590-9230~9234
Fax: +86-10-6590-9235

Anritsu Corporation Shanghai Liaison Office

No.511 Shanghai Jing Tai Building
No.58, Mao Ming Nan Rd. Sanghai
200020 P.R. China
Phone: +86-21-6415-5137
Fax: +86-21-6472-6677

Anritsu Company Ltd Guangzhou Representative Office

Room 720, 7/F., Dongshan Plaza,
69 XianLie Road Central.
Guangzhou 510095 P.R. China
Phone: +86-20-8732-2231
Fax: +86-20-8732-2230

Anritsu Corporation New Delhi Liaison Office

Room No. 508, Prakash Deep,
7 Tolstoy Marg, New Delhi-110001, India
Phone: +91-11-331-9133
Fax: +91-11-371-3948

Associated Electric Trading Corp.

Zia Chambers, 25 McLeod Rd., Lahore,
Pakistan
Phone: +92-42-722-1716
Fax: +92-42-7221456

Chris Radiovision Ltd.

Kouloumbris Building, 23 Crete Street,
P.O. Box 1989, Nicosia, Cyprus
Phone: +357-2-466121
Fax: +357-2-365177

Electronic Equipment Marketing Co.

P.O. Box 3750, Riyadh 11481,
Saudi Arabia
Phone: +966-1-4771650
Fax: +966-1-4785140

Etecsa (Pty) Ltd.

1st Floor Montrose Place, Waterfall Park,
Bekker Rd., Midrand, South Africa
Phone: +27-11-315-1366
Fax: +27-11-315-2175

Giza Systems Engineering

2 El Mesaha Square, Dokki A.R.E.,
P.O.Box 1913, Cairo 11511, Egypt
Phone: +20-2-349-0140
Fax: +20-2-360-9932

Infotechs Ltd.

23-1, Jaya Rd., Colombo 4, Sri Lanka
Phone: +94-1-580088
Fax: +94-1-584644

Inter Muhendislik Danismanlik ve Ticaret A.S.

Farabi Sokak No: 24/14
Cankaya-06680 Ankara, Turkey
Phone: +90-312-4277792
Fax: +90-312-4277937

Jasmine Telecom Systems Co., Ltd.

333 Laksi Plaza 6th Floor, Tower 2,
Chaengwatana Rd., Donmuang, Bangkok
10210, Thailand
Phone: +66-2-576-0200
Fax: +66-2-576-0420

Meera Agencies (P) Ltd.

A-23 Hauz Khas New Delhi 110 016, India
Phone: +91-11-685-3959
Fax: +91-11-685-2275

UAE Utmost Electronics Trading (L.L.C.)

P.O. Box 41175 Abu Dhabi, U.A.E.
Phone: +971-2-768909
Fax: +971-2-768907

Martwell Electronics Pvt., Ltd

3rd Floor, Francis House, Stanley Avenue,
P.O. Box 1737, Harare, Zimbabwe
Phone: +263-4-793578
Fax: +263-4-737956

Mandeno Electronic Equip. Co.

463 Mt. Eden Rd. Mt. Eden,
Aukland 1003, New Zealand
Phone: +64-9-630-7871
Fax: +64-9-630-1720

National Projects and Technology Co. L.L.C

P.O. Box 97, Wadi Al Kabir, Postal Code 117,
Sultanate of Oman
Phone: +968-791704
Fax: +968-791697

O'Connors Engineering & Trading (Malaysia) Bhd

3rd Floor, Wisma Siong Huat,
Lot 13, Jalan 51A/223
46100 Petaling Jaya,
Selangor Darul Ehsan, Malaysia
Phone: +60-3-757-2828
Fax: +60-3-757-7871

P.T. Subur Sakti Putera

Jalan Musi No.32, Jakarta 10150,
Indonesia
Phone: +62-21-3803644
Fax: +62-21-3845043

Qatar Communications Ltd.

P.O. Box 2481, Doha, Qatar
Phone: +974-424347
Fax: +974-324777

Trading and Agency Services

P.O. Box 1884, Doha, Qater
Phone: +974-432212
Fax: +974-422255

Rajab & Silsilah & Co.

P.O. Box 203 Jeddah 21411, Saudi Arabia
Phone: +966-2-6610006
Fax: +966-2-6610558

Salritsu International Trading Corporation

50B ODC International Plaza
Condominium, 219 Salcedo St., Legaspi
Village, Makati, Metro Manila, Philippines
Phone: +63-2-816-2646
Fax: +63-2-815-0986

Sedel

24, 26, Bd, Resistance, Casablanca,
Morocco
Phone: +212-2-302444
Fax: +212-2-449311

Superior Electronics Associated

B-98 Block H, North Nasimabad, Karachi
33, Pakistan
Phone: +92-21-613655

Tareq Company

P.O. Box 20506 Safat, 13066 Safat, Kuwait
Phone: +965-2336-100
Fax: +965-2437-700

Tech-Cent Ltd.

Haarad St. No.7, Ramat Haahayal,
Tel-Aviv 69710, Israel
Phone: +972-3-6478563
Fax: +972-3-6478334

Test

Sehit Adem Yavuz Sokak No.6/17,
Kizilay-Ankara, Turkey
Phone: +90-41-71086
Fax: +90-41-74384

Japan

Head Office

5-10-27, Minamiazabu, Minato-ku,
Tokyo 106-8570
Phone: 03-3446-1111
Fax: 03-3442-0235

Atsugi Factory

Measurement Solutions

1800, Onna, Atsugi-si, Kanagawa 243-8555
Phone: 046-223-1111
Fax: 046-225-8379

If you have any comments on this manual, please e-mail them to Manual-support@zz.anritsu.co.jp.

www.manualsplus.com
1030568